



Technical Report GriPhyN-2001-xx
www.griphyn.org

GriPhyN Overall Project Plan

Version 14
22 December 2001

Developed by members of the GriPhyN Project Team

Submit changes and material to:

Mike Wilde, editor
wilde@mcs.anl.gov

Table of Contents

1	Introduction: Managing GriPhyN	3
2	The GriPhyN Vision	6
3	The GriPhyN Computer Science Research Program	8
4	Research Milestones	9
4.1	Virtual Data	9
4.2	Request Planning	10
4.3	Request Execution	11
5	VDT Milestones	12
6	Infrastructure (testbed) construction.....	14
7	Project Process Flow	16
7.1	Application analysis	17
7.2	Challenge Problem Identification	19
8	Coordination Between Grid Projects.....	20
8.1	Coordination Regarding Virtual Data Toolkit	22
9	Project Logistics.....	22
9.1	Coordination Meetings	22
9.2	Communications.....	22
9.3	Planning.....	23
9.4	Reporting.....	23
9.5	Project Personnel	23
9.6	Faculty	24
9.7	Project Team Structure	25
10	Education and Outreach	25
10.1	Web page for GriPhyN E/O.....	25
10.2	Research Experience for Undergraduates (REU) supplement	26
10.3	Grid-enable the UT Brownsville Linux cluster	26
10.4	Involving other minority serving institutions	26
10.5	Leveraging on-going existing E/O programs.....	26
10.6	Course development	27
10.7	Workshops and tutorials	27
10.8	Other activities.....	27

1 Introduction: Managing GriPhyN

The goal of GriPhyN is to increase the scientific productivity of large-scale data intensive scientific experiments through these Grid-based approaches:

- Apply a methodical, organized, and disciplined approach to scientific data management using the concept of virtual data to enable more precisely automated data production and reproduction.
- Bring the power of the grid to bear on the *scale* and *productivity* of science data processing and management.

Virtual data is to the Grid what object orientation is to design and programming. In the same way that object orientation binds method to data, the virtual data paradigm binds the data products closely to the transformation or derivation tools that produce data products. We expect that the virtual data paradigm will bring to the processing tasks of data intensive science the same rigor that scientific method brings to the core science processes: a highly structured, finely controlled, precisely tracked mechanism that is cost effective and practical to use.

Similarly for the Grid paradigm: we see grids as the network OS for large-scale IT projects. Just as today the four GriPhyN experiments use an off-the-shelf operating system (Linux, mainly), in the future, our goal is that similar projects will use the GriPhyN VDT to implement their Grid. The road to this level of popularization and de facto standardization of GriPhyN results needs to take place outside of and continue beyond the time frame GriPhyN; hence the partnerships that we will create between GriPhyN and other worldwide Grid projects are of vital importance.

One of GriPhyN's most important challenges is to strike the right balance in our plans between research – inventing cool stuff – and the daunting task of deploying that “stuff” into some of the most complex scientific and engineering endeavors being undertaken today. While one of our goals is to create tools so compelling that our customers will beat down our doors to integrate the technology themselves (as they do with UNIX, C, Perl, Java, Python, etc), success will also require that we work closely with our experiments to ensure that our results do make the difference we seek.

We will be successful if enough results of GriPhyN flow into the 4 experiments to make a difference for them, and demonstrate the value and future of continuing on this path to make a high-value contribution to enhance the ability of science to deal with high data volumes efficiently, cost effectively, and thus open new doors.

Achieving these goals requires diligent and painstaking analysis of highly complex processes (scientific, technical, and social); creative, innovative, but carefully focused research; the production of well-packaged, reliable software components written in clean, modular, supportable code and delivered on an announced schedule with dependable commitment; the forging of detailed integration plans with the experiments; and the support, evaluation, and continued improvement and re-honing of our deployed software.

While GriPhyN is a research project, its scale and importance and its complex relationships with other projects makes it important to identify clear goals, milestones, and schedules, both for internal planning and for use by our external collaborators. This document, which we revise periodically over the course of the project, provides the highest level view of this information, and serves as a master-plan for the project. Its scope includes all activities that are common to all four of the participating science experiments. To supplement the master plan, all activities that are specific to the GriPhyN interaction with each experiment are described in a planning document for that experiment. These planning documents each cover one project year, running from October 1 to September 30.

Some of the work to be undertaken by GriPhyN is being performed in collaboration with participants in PPDG, EU Data Grid, and DOE SciDAC, as well as, of course, the four physics experiments with whom GriPhyN is partnered. We indicate what components or services we expect to obtain from these projects, and appropriate contingency plans if these deliverables are not forthcoming. We are also working in partnership with the NMI GRIDS Center to provide support for our VDT, and with the iVDGL project to create and operate testbeds.

The GriPhyN methodology for making this difference involves the integration of five overlapping top-level processes: research, development, integration, support, and evaluation.

Research – exploring current knowledge and results; proposing new paradigms and techniques; documenting new architectures; building prototypes of new frameworks and architectures; evaluation of the prototypes; research also includes the “market analysis” – the detailed study of our customers, the four physics experiments.

We need to strike a balance between focusing research to solve specific problems of the experiments, and letting research come up with new techniques beyond those that the experiments can even envision today.

Development – taking results from research and turning them into packaged software components that can be readily delivered to and installed by customers. Our plans in this area are described as successive releases of the GriPhyN VDT, as described in Section 5, VDT Milestones.

Integration – the process of planning and executing the enhancement of experiment data processing by integrating GriPhyN components into the experiments IT. We are working closely with the experiments to execute this step, as described in the individual experiment plans.

Support – in order for our tools to be used in such serious scientific projects, they need to be highly reliable, supportable, and *supported*. Since GriPhyN clearly has limited resources, we need to forge relationships whereby support comes largely from Grid support projects like the NMI GRIDS Center, and from the customers themselves. This comes back to the requirement to create tools that are reliable and require minimal support; to document tool usage clearly but at lowest possible cost; and to leverage and enlist the support of the user community itself to support and contribute to the toolkit.

Evaluation – in order to succeed, we must continually, diligently, and critically evaluate our software tools, with a critical eye to their deficiencies even as we promote them based on their strengths and benefits. Support and evaluation processes are closely coupled, as we can learn the most about our tools weaknesses when we work closely with the integrators and users of the tools. If we lose sight of this while focusing on ongoing research, we will not succeed. Thus, our project plan includes regular “challenge problems” (the first of which have already been completed) to support evaluation and feedback back into our research and development processes. Some deficiencies will be the result of development shortcomings, while others will dictate that we go back to the research drawing board and look for better ways to solve customer problems.

We view this process as a pipeline (albeit with numerous feedback paths). Not all research results make it into live use, but all are evaluated at some scale.

The critical computer research breakthroughs we are pursuing to achieve these goals are in the areas of:

- The virtual data paradigm, and its supporting catalog structures and integration languages

- Policy and condition-sensitive execution planning and scheduling algorithms and architectures
- Ubiquitous, globally accessible, highly available cataloging systems
- The Petabyte range scalability of data storage and transport and cataloging systems
- Interfaces and levels of automation that make a worldwide grid as easy to use as a workstation

Of equal importance are the critical engineering and project management capabilities we need:

- The ability to turn research results into robust and supported tools that can gain widespread adoption
- The ability to design tools that empower scientific software developers and capture their imagination
- The ability to thoroughly analyze scientific data processing paradigms and uncover and simplify data dependencies
- The ability to understand and overcome the social and organizational issues that often block the adoption of off-the-shelf software by large, complex projects

In the rest of this document, and in a set of four subsidiary “Application Plan” documents, we provide both background information on GriPhyN and detailed technical roadmaps for the various components of the project: computer science research, virtual data toolkit development, and application integration. Because the technical landscape in which we operate is so complex, and the interrelationships between GriPhyN and other activities (ATLAS, CMS, LIGO, SDSS, NVO, EU DataGrid, PPDG, iVDGL, TeraGrid, etc.) are so critical to planning, we focus our planning efforts on identifying:

- The technology development and application integration tasks to be undertaken during the next 12 months, for which we provide detailed plans; and
- The research priorities for CS research that will address what are seen as likely stumbling blocks in out years.

We do not attempt to provide detailed task lists for more than 12 months out, but instead work constantly to update our 12-year-out plans in the light of progress to date and our assessment of the evolving external situation. However, we can express in general terms how we expect GriPhyN to progress over years 2-5 of the project.

In Year 2, as described in considerable detail elsewhere in this document and in the four Experiment Plans, we will deploy the VDT with non-distributed virtual data support; deploy first planner and policy language; integrate virtual data into real efforts in each experiment; start research foci in planning, fault tolerance, and knowledge representation; and demonstrate scaling to hundreds of processors and O(100 TB) of data.

In Year 3, we will introduce distributed, scalable, and fault tolerant virtual data catalog services; deploy scalable and fault tolerant execution services; start research foci in knowledge representation for virtual data; execute substantial challenge problems in each experiment; and demonstrate scaling to thousands of processors and O(1 PB) data.

In Years 4 and 5, we will undertake first field tests and then deployments of knowledge representation techniques and undertake substantial international challenge problems (and, we would expect, production computations). During this time, we would also conduct considerable tuning and evaluation, move forward to new versions of planning and catalog structures, work to

deploy VDT and its various components widely, including to non-GriPhyN experiments, and provide support for VDT users.

2 The GriPhyN Vision

The science projects that we target share the common need to harness large-scale distributed resources through data grid technologies. We state an approach to doing this by describing what the four GriPhyN experiments should look like when the fruits of the project in place.

This vision has a direct bearing on our project planning effort. If the scenarios described below depict the end goals of this project, we must create a year-by-year plan that clearly identifies how we'll develop the specified capabilities. This will demand a lot of inter-related and inter-working technologies and components, and will require that we solve research problems in a manner that creates solutions for the missing pieces of this puzzle.

Each step in our plan needs to fit clearly into building the type of solutions that we have described in the following scenarios.

Scientists can seamlessly harness powerful grid resources across multiple organizations with little knowledge of the complexities of resource allocation and distributed computing.

Example: a CMS physicist can look in a catalog for simulation results. Some of the desired results might be already at their site; others may be at other sites and can be fetched quickly. Still others existed at one time and can be re-derived; the remote network to yet another set of results is going to be congested with a major transfer for the next 8 hours, so a new computation is kicked off to re-derive some of these results, which will finish in 1 hour. The new computation uses 75% local resources, the remaining resources are from remote sites with available cycles on uncongested network paths.

The analysis job that needs to run in these results is scheduled and initiated when all data dependencies have been located or materialized. This job runs at 4 different sites, and the final result is emailed to the scientist in the morning. The scientist can check status of the computation at any point, can stop or pause the job; sometimes even steer it.

Despite the scale and the complexity of the resources used here, to the physicist this task was no more difficult than if all the work was done on a laptop – the Grid was as easy to use as a PC.

Experiment data is tracked in a uniform manner, clearly identifying how most data objects were derived.

Example: A scientist questioning the validity of an analysis can look in the catalog, find that the analysis was based on 1000 event reconstructions, and can check which version(s) or reconstruction code was used to create each of the 1000 events. She discovers that 15 events were reconstructed using outdated code, and she initiates a new reconstruction for these events, keeping the new data in a private store. She then notifies her data administrator of the problem, pointing him to the new events; the data administrator then replaces the outdated reconstructions, and interrogates the virtual data catalog to look for similar events that require upgrading – anywhere in the collaboration, anywhere in the world. With a simple change to a data derivation specification, the results are recomputed, much like a complex program is rebuilt with a simple invocation of a “make” command.

Resource allocations are controlled, measured and tracked by resource administrators who set policies to achieve and arbitrate the overall goals of both the experiment's virtual organization and the resource owners.

These policies are not excessively complex to express and maintain, and they control the way in which the grid machinery executes user requests. Resource policies are used to control the use of storage, computing, and network resources by users, groups, and virtual organizations.

Enormous resources can be brought to bear with relative ease, but in a controlled fashion.

*Example: When the ATLAS data administrator at CERN has received the final approval from the developers of a 5 stage pipeline of reconstruction software (who work in Italy, Scotland, France, Moscow, and California), he initiates the rebuilding of the Event Summary Data of **4 petabytes** of raw data. This process takes 5 weeks to complete, and uses resources in 25 centers of 8 countries. 4 of these centers were not even part of the ATLAS collaboration when the reconstruction run started.*

As the run proceeds, most failure scenarios are handled automatically and transparently. Failing compute nodes are brought out of service and reported for repair through grid and cluster monitoring and management frameworks. Failing jobs are restarted automatically, many from checkpointed results. The progress of the run is continuously monitored and reported back to the group of the submitting administrator. The success of the run is independent of not only the client computer from which it was submitted, but in fact from the failure of the entire site of the submission.

Scientists use end-user-oriented tools and express their jobs in science terms rather than in CS terms. *Scientists can say: I want to run code X on data Y, or even more simply: I want data object Z. The automated grid “planner” mechanism decides where to get the data from, where to run the code, when to run the code, and tells the user the expected completion time. Then the planner decides where to place the resulting data. The planner can slow down, pause, or stop existing work if the new work has sufficiently high priority. The planner (if asked) can explain its decisions to the user, much like today’s SQL query optimizers can diagram a query plan and trace how various query alternatives were selected. The execution planning decisions factor in both site-specific policies and dynamic resource utilization levels.*

Scientists request data and/or computations using powerful job description languages, with a high degree of interoperability with visual tools. The right balance of visual and textual tool usage is made possible through well-designed architectures. These languages help expose parallelism, fault recovery, and data dependencies and derivations and enhance the location-independence of job specifications. Scientists can use simple but powerful job control languages, embedded in familiar scripting languages, to request data products and transformations.

For example, an astrophysicist in Moscow uses a graphical user interface to simulate the gravitational waves produced by a collapsing black hole, running a multi-stage pipeline to filter information in both the time and frequency domain. She then asks the interactive science portal to emit the job control code for the pipeline, and using a simple Python script creates the virtual data definitions in the project’s centralized data catalog to define the output of 5000 such analysis pipelines on a large set of input files from the gravitational wave observatory. Colleagues at Caltech and MIT then explore this virtual data, each requesting different 3000-element subsets of the virtual datasets. These overlapping products result in 4000 pipelines, executing at 5 sites around the world, totally transparently to any of the scientific collaborators.

Users are given performance predictions for their jobs before they submit them, with alternatives spelled out for them so that they can make prudent cost-delay-benefit decisions. *In cases where extremely complex jobs are being scheduled, potentially taking days or weeks to fully complete, the users responsible for these jobs can experiment with policy alternatives to guide data sources or data placement, or priorities for computing resources, to achieve the*

necessary goals. In addition, users can select tradeoffs in execution plan based on quality of input sources and/or transformation algorithms (e.g., fast and accurate vs. slower but lower resolution simulations).

Such long running jobs are amazingly robust and fault tolerant in their execution, working around downed hardware, unavailable data, space shortages, and network outages, and re-adjusting the execution plan as needed to continue with little or no manual intervention and updated completion estimates.

The GriPhyN goal is to create unified, software components and toolkit solutions that are used *in common* across all four experiments, and that these results are used by future experiments in other disciplines with relative ease, changing and enhancing the way data-intensive science will be done in the future.

3 The GriPhyN Computer Science Research Program

GriPhyN's success depends on achieving two fundamental breakthroughs in computer science research. Our premises are:

- 1) that we can develop a paradigm, framework, and representation for the complex dependencies between science data objects, and can efficiently and with high integrity capture and accurately replay with high fidelity the steps needed to re-derive data.
- 2) that we can abstract and automate the highly complex and policy-driven decision making processes to automatically schedule work in a complex grid of loosely coordinated resource pools without central ownership.

The computer science research program that we have created to achieve these capabilities is organized as follows:

Institution	Research Topics
University of California Berkeley and LBL	Database query processing research; Request management; Tertiary storage management; simulation
University of California San Diego	Fault tolerance; Metadata management; Storage resource management architecture (SRB)
University of Chicago	Virtual data cataloging and processing; domain knowledge representation; policy research; distributed catalog architectures; simulation of replica placement for computational proximity
Indiana University	User management; Science Portals; portal-to-grid interface language; Grid Operations software
Northwestern University	Performance data collection, analysis and predication; Infrastructures for monitoring and measurement
University of Southern California Information Sciences Institute	Virtual data cataloging and processing; Execution planner algorithms; Request management components; Fault Tolerance; distributed catalog architectures
University of Wisconsin	Job description language and scheduling algorithms and frameworks (Condor, ClassAds, Matchmaking, DAGMan); storage appliance and integration of data movement into job descriptions and processing; recovery of data transfer (NeST

	and Kangaroo)
--	---------------

Cross-institution collaborations currently taking place within the teams above include:

Virtual Data cataloging and processing (UC & ISI)
 Robust distributed scalable cataloging service architectures (UC & ISI)
 Fault Tolerance (UCSD, UW, and ISI)
 Virtual Data Grid Simulation: (UC and LBL)

As can be seen from this list, GriPhyN will provide an extremely rich and fertile ground for the selection of a large number of doctoral-level thesis topics.

4 Research Milestones

The following list, organized by technology area, describes the major milestones in the project for addressing the key technology deliverables that will then be packaged into the VDT and made available for integration into the experiments.

The details of the integration and deployment efforts are described in the detailed project plans being developed by GriPhyN sub-teams working directly with the experiments. (The year-2 plans for these efforts are now available in separate documents temporarily posted at www.mcs.anl.gov/~wilde/griphyn).

4.1 Virtual Data

Year 1

- Develop basic information model to represent data elements, the relationships between different data types and the characteristics of data elements. Develop protocols for storing, discovering and retrieving these models. Design and develop tools for creating, accessing and manipulating these models by interactive tools, and planning and scheduling tools.

Status: accomplished; demonstrated at SC2001

- Deploy centralized metadata and replica catalog services. Develop tools for managing catalogs.

Status: All four experiments have progressed with their own metadata databases. ATLAS and CMS (via the EUDG) have deployed the Globus replica catalog in their testbed. GDMP, used within that testbed, is based on this catalog implementation.

Year 2

- Develop techniques for representing data transformations, and integrate these techniques into the information model. Develop methods and catalogs for categorizing and curating code elements.

Status: Virtual Data Catalog version 0, and a corresponding manipulation language, Virtual Data Language 0, was developed and demonstrated at SC2001. The design for VDC/VDL1, a candidate for VDT release 2.0, is scheduled to begin in Jan 2002.

- Extend catalog services to support distributed and replicated catalogs. Develop techniques for failure detection and fail-over in the situation of catalog failure.

Status: A design for a robust and scalable distributed catalog (the Replica Location Service) has been designed and documented, and is circulating for review among the GriPhyN (Globus) and EUDG teams. Prototyping of that design has begun.

Year 3

- Extend information model to support multiple versions of both data dependencies and data transformation components. Also extend catalogs to support interfaces to request planning and request execution modules.
- Develop distributed algorithms for discovery of information across distributed virtual data catalogs.

Year 4

- Augment the information model to include information about alternative implementation of data transforms with alternative performance characteristics.
- Develop methods for collecting historical performance information and incorporate into the catalogs.

Year 5

- Augment information model to incorporate local and global policy constraints.

4.2 Request Planning

Year 1

- Develop generic models for representing execution plans. Define a set of API and tools for constructing, traversing, and manipulating plan data structures. Develop protocols and formats for storing and exchanging execution plans.

Status: DAGman – the directed acyclic graph manager, has provided the project with a unifying language to manipulate and communicate grid execution plans.

- Develop uniform policy representation for code, data and resource access. Develop a set of global and local policy scenarios that reflect the requirements of the user communities of the four physics experiments.

Status: The Community authorization server has been developed and is scheduled for incorporation into the VDT 2.0 release. Research into policy language has begun at the U of Chicago.

- Develop simple optimization heuristics. Initial thrust will be on data movement only and focus on the use of alternative, or branching plans to compensate for both resource failure and changes in resource performance. Implement planning heuristics in prototype planning module. Evaluate performance of alternatives with simulation and model based studies, as well as execution on GriPhyn testbed.

Status: Basic planning manipulation, and the transformation of abstract to concrete execution plans has been prototyped and demonstrated for both CMS and LIGO at SC2001.

Year 2

- Design a basic planning API, to facilitate access to remote planning services from high-level tools without dependence on underlying planning heuristics or planning methods. Define and implement planning toolkit, providing access to catalogs as well as remote planning servers.

Status: DAGMan has provided a start in this area.

- Extend planning toolkit to incorporate global and local policy considerations into policy construction. Initial focus will be on the application of matchmaking as a means method for the introduction of policy.

Status: research in progress on this at U of Chicago.

- Extend optimization heuristics to include computational resources and data transformations (i.e., code). Evaluate the use of alternative plans to meet optimization goals.

Year 3

- Extend request planning APIs and toolkit to support incremental plan generation and dynamic replanning. This extended interface will be used to couple the request planner with the request execution services.
- Extend the range of planning algorithms to incorporate alternative optimization heuristics, for example including factors such cost.
- Investigate the hierarchal and distributed planning algorithms and evaluate their impact on scalability, reliability, and the ability to share plans across multiple, independent requests.

Year 4

- Develop algorithms that incorporate policy constraints into request planning process. These algorithms just examine the constraints applied to each element of the request being planned, and respect the constraints for each local resource as well as for the entire request, with respect to global policies. Initial focus will be on static, non-incremental planning.

Year 5

- Extend policy sensitive optimization algorithms to incorporate incremental planning. Develop hybrid strategies that combine static and incremental planning. Evaluate performance of new planning algorithms both in simulations and on testbed.

4.3 Request Execution

Year 1

- Develop and evaluate a task control language capable of capturing the requirements, preferences and dependencies of a PVDG request. Implement prototype of an interpreter to a basic subset of the language.
- Develop a protocol for information exchange between the execution and planning agents.

Status: DAGMan graphs (“abstract” and “concrete”) have been adopted for both of these purposes and used in prototype demos.

Year 2

- Develop an execution agent capable of receiving a simple plan from the planner and interacting with the PVDG services and resources in order to carryout the plan.

Status: The integration of DAGMan, Condor, and the Condor-G agent has been demonstrated to accomplish this.

- Develop a protocol for the exchange of co-allocation information (availability, policy, statistics, ...) between the planner and the co-allocation agents.

- Develop a basic portable and configurable event and trigger manager.

Status: being explored by the joint PPDG-GriPhyN Monitoring working group.

MDS2

- Develop a framework for gathering statistics on the resource consumption profile of completed and in-progress requests and the availability of resources.

Status: being explored by the joint PPDG-GriPhyN Monitoring working group.

Sudharshan, MDS2 , GridFTP perf eval

Year 3

- Develop a fault-tolerant version of the execution agent.

Status: research in progress at UCSD.

- Develop a basic recoverable co-allocation agent. The agent will support basic reservation services.
- Add fault-tolerance to the Condor master process (Keith Marzullo at SDSC) and reliability to basic propagation protocols.

Note: DAGman stuff has fault tolerance

- Develop a fault-tolerant and persistent repository of PVDG statistics

Year 4

- Develop a distributed (mobile) version of the execution agent and enhance the ability of the agent to adapt to changes in the availability, location and capabilities of the grid resources.
- Interface co-allocation agents with planning agents.
- Develop reliable, efficient and secure event propagation and notification protocols
- Develop and implement dynamic and incremental execution algorithms

Year 5

- Evaluate the performance of different execution policies.
- Evaluate co-allocation and reservation policies.
- Add real-time services to the event and trigger system.
- Evaluate impact of incremental and dynamic planning on request execution.

5 VDT Milestones

The following release plan summarizes our initial vision of the main features delivered each year. We will produce one major release of the VDT every project year. VDT features will be enhanced and refined from experience and user feedback, with enhancements introduced throughout each project year through point releases. Note that this integration and deployment activity tracks the principal features listed earlier in the CS research program description. *Also note that this schedule of feature rollouts is still subject to change, based on the availability of*

underlying technology and the changing needs of experiment deployments and integration.

The first release of VDT as a integrated whole will be packaged and deployed in testbed efforts in PY2-Q2. All of the components of VDT-1 are already in use in the USCMS and USATLAS testbeds.

VDT 2 will be the first release that contains a virtual data catalog. This will be a version based on the VDC that was demonstrated at SC2001, with features from the LIGO and CMS prototypes consolidated, and significantly enhanced based on early experience and further CS research design effort. We expect to deliver VDT 2 first release in the PY2-Q3 with several point releases of 2.0 before end of PY2.

Once VDT 2 is released, subsequent releases will be scheduled for the second quarter (Jan-Mar) of the remaining 3 project years. The rationale for this is to synchronize the yearly project cycle with the annual Supercomputing conference, which takes place each fall at the beginning of the GriPhyN project year. This conference is an excellent forum in which to showcase GriPhyN results, and from which to take back a lot of useful hands-on experience from building prototypes and demos that can guide the release of subsequent VDT components.

The overall VDT plan is as follows:

VDT 1.0 (Basic Grid Services) provides an initial set of grid enabling services and tools, including security, information, metadata, CPU scheduling, and data transport. VDT-1 will support efficient operation on O(10 TB) datasets, O(100) CPUs, and O(100 MB/s) wide area networks and will build extensively on existing technology.

ClassAd toolkit (library)

Condor 6.3.1

DAGMan

Globus 2.0

GDMP 2.0

VDT 1.0 will also include: configuration scripts for replica catalog, GridFTP, GDMP, and MDS that are specific to the GriPhyN / iVDGL test grids. It is tentative slated to use the PACMan installer to provide easy and uniform installation across all GriPhyN testbeds.

Tentative release date: PY2-Q2

VDT 2.0 (Centralized Virtual Data Services) provides a *first set of virtual data services and tools*, including support for a centralized virtual data catalog, centralized request estimation, centralized request planning, network caching, and a simple suite of distributed execution mechanisms. Representation and exchange of local policies will be supported for network caches.

Potentially includes:

Virtual Data Catalog structures and VDL engine

VDL and rudimentary centralized planner / executor

Community Authorization Server

Initial Grid Policy Language

User login management tools

Tentative release date: PY2-Q3

VDT 3.0 (Distributed Virtual Data Services) supports *decentralized and fault tolerant execution and management* of virtual data grid operation, via integration of distributed execution mechanisms able to select alternatives in the event of faults, agent-based estimation and monitoring mechanisms, and iterative request planning methods. This will be a major release,

and will depend on new functionality from Globus, Condor, and other sources. This version will support O(100) TB datasets, O(10 TB) network caches, O(1000) CPUs, and O(400 MB/s) networks.

Potentially includes:

- Reliable File Transfer service
- Striped GridFTP service
- Managed storage elements (NeST-based)
- Policy-based planner
- Enhanced monitoring and measurement infrastructure
- Distributed high-capacity catalogs (based on Replica Location Service)
- Virtual data generation semantics
- Basic fault tolerance (master-worker fault tolerance for Condor)
- Metadata database integration (from SRB)

Tentative release date: PY3-Q2

VDT 4.0 (Scalable Virtual Data Services) *scales virtual data grid operation to realistic magnitudes*, supporting applications involving widely distributed O(1 PB) datasets, O(100 TB) network caches, and O(10,000) CPUs.

Potentially includes:

- Science Portal interfaces
- Distributed planner
- Performance measurement and prediction framework
- "Fuzzy" virtual data description mechanisms
- Catalog and Transport scalability enhancements
- Advanced fault recovery

Tentative release date: PY4-Q2

VDT-5 (Enhanced Services) enhances VDT functionality and performance as a result of application experiences.

The final project software deliverable. Provides an enhanced and stable VDT base and is packaged for general use outside GriPhyN, in particular for use by other scientific disciplines.

Potentially includes:

- Knowledge representation framework (ontology-based) for the virtual data catalog
- Object tracking for OO and relational databases
- Additional planner transparency and sophistication
- Further enhancements in fault tolerance, robustness, and transparent recovery

Tentative release date: PY5-Q2

6 Infrastructure (testbed) construction

The computing infrastructure of GriPhyN will involve three levels of Grid resources:

Research testbeds: small grids where software or application research can be conducted and tools developed. These testbeds will consist of Linux platforms of diverse releases, representing what is currently needed to support experiment application code. This grid, however, will be made available to the entire GriPhyN community.

Experiment testbeds: larger grids, owned by each experiment, where challenge problem solutions can be developed and their feasibility measured and demonstrated. These grids exist today, in the form of the US ATLAS test grid, the prototypical US CMS and LIGO grids that were used at SC2001.

Production resources: where challenge problem solutions can be placed in live (production) use by the experiments. These grids are being planned and constructed by the iVDGL and the LHC Computing Grid projects.

The current vision is that these resources will be unified into a single “GriPhyN Grid” to provide uniform access to and control of these resources as needed by the project. Currently, we expect this grid to contain some mix of research, challenge problem, and a limited amount of production testbeds within it. (We defined a “testbed” as a set of nodes within the Grid).

The grids listed here will:

- run successive releases of the VDT
- be available to all or selected GriPhyN project members for research, development, and challenge problem work
- utilize a single Certification Authority
- have well-maintained Grid Information Services
- contain an agreed-upon mix of job execution facilities
- contain other shared infrastructure such as test and production replica location services and data transfer services

The experiment-maintained testbeds (for example, the ATLAS testbed) would be used mainly in the challenge-problem development and demonstration phases of the project, and the construction and management of those testbeds could be handled by the experiments and related projects (such as PPDG in the case of ATLAS and CMS).

Most of the final stage of GriPhyN solution development – live deployment – is expected to take place on the experiment’s production resources, but we expect that there may be cases where some types of live deployment can take place on GriPhyN grids (for example, running preliminary analyses where the experiment does not require complete control of the execution environment).

Compatibility issues: if challenge problems are developed using a specific VDT toolset, it’s important that deployment take place on an identical or compatible base. Due to the complexity of the tools (both grid and application) involved, it’s very difficult to ensure that GriPhyN-developed solutions will run correctly if the target environment is not precisely matched to the development environment.

The grid testbed construction tasks will include:

- Identification of resources (hosts, storage servers, networks)
- Design of login administration mechanism and certification mechanism
- Installation of application software
- Installation of VDT
- Creation of VO’s
- Establishment of CAS’s and policies (policy design a major task)

- types of work
- types of user groups
- priorities of access to resources: computing, storage, network
- Design and setup of Catalog architectures
- Namespace management

Once the basic infrastructure is in place for both production and research, the level of infrastructure effort should diminish somewhat, and involved mainly the installation of new releases of the VDT.

In our process diagram (figure 1), infrastructure deployment is shown keyed to the availability of new toolkit releases; note that it could also be triggered by the availability of new hardware resources that could be integrated into the GriPhyN testbed.

7 Project Process Flow

Although the complex worlds of the four GriPhyN experiments defy common and uniform solutions, we nonetheless propose to base the GriPhyN activities within each experiment on a similar planning approach, based on 1-year cycles. This section describes this common, and how the experiment activities interplay with CS research and toolkit development.

This yearly goal-oriented approach will fit well with cyclic events such as project reviews and the demo-driving Supercomputing conference; it may require adjustments, however, in order to accomplish integrations of deliverables into experiment plans that are each driven by a project-specific calendar.

The figure below describes an idealized one-year activity plan for the overall project; for each remaining year we would presumably follow a similar pattern (at least, at this point in our planning). CS activities are shaded dark, experiment activities light. The activities nearer the top of the figure feed the activities lower down, with the challenge problem solutions representing the ultimate GriPhyN goals. A very important aspect of our coordination plan is that the CS activities *span across experiments*, striving to conduct research and create tools that fit the needs of *all* of the experiments.

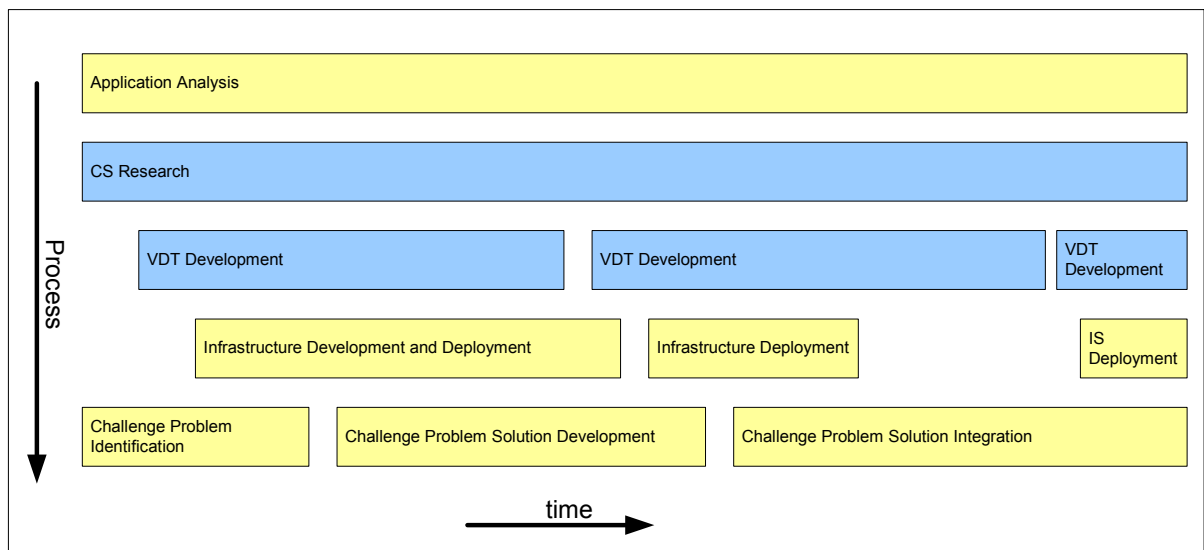


Figure 1: Common Yearly Plan for Experiment Activities.

The yearly plan predicts about 2-3 VDT point releases per year (based on a VDT major-release plan described in Section 3), continuous CS research and application analysis (the latter at a steady but less intensive rate), and one cycle of challenge problem identification, development, and integration. As appropriate, the challenge problem cycle can be repeated several times per year, possibly in an overlapped fashion, depending on the nature of the chosen problems and available integration opportunities that are driven by experiment needs and schedules. This plan reflects the project interaction model that was described in the original proposal, shown in figure 2, below.

The main activities of this common planning approach are outlined in the following sections.

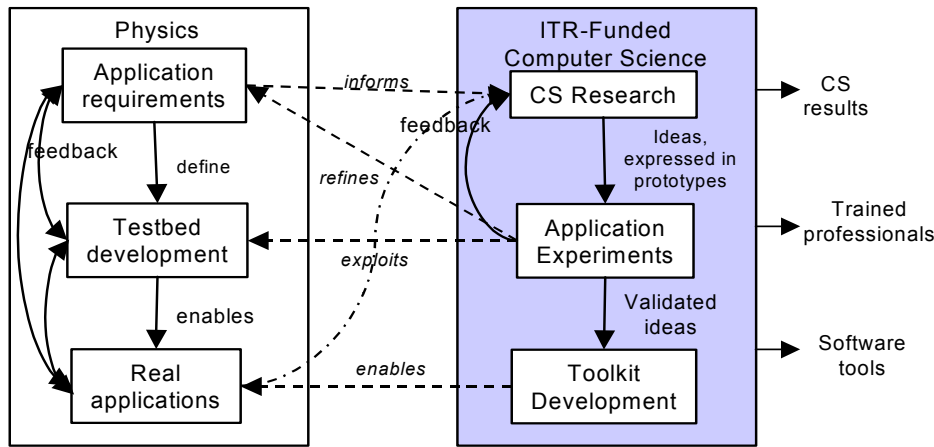


Figure 2: Process and information flow within the GriPhyN project.

Describe seed planting paradigm: vdl; measurements;

7.1 Application analysis

The purpose of application analysis is to determine what processes in each experiment could benefit from the type of results that GriPhyN seeks to produce, to refine the requirements for GriPhyN research, development, and deliverables, and to figure out how to apply the results back into the experiment. *This activity is critical to the relevance and utility of GriPhyN results to the experiments.*

Unfortunately, this activity has also proved to be difficult to conduct, and has not yet yielded the necessary information back into the project. Difficulties include: much of the information is not yet known, and needs to be extrapolated from past experience; experiment architectures, applications, and decisions are still being formed; the information involved is heavily distributed and the project documents and information sources in which requirements are embedded are usually voluminous. On the other hand, the LHC experiments have been very successful at modeling the expected processing flows within their applications, so there is a positive basis for hope here. We propose to re-assess the process and to find an approach that works and delivers the expected benefits to the project.

Another goal of application analysis is to compare requirements of the four experiments, and then identify common needs that can be met with common tools, architectures, paradigms, testbeds, and infrastructure. If the mechanisms that GriPhyN seeks to create need to be heavily customized for each experiment, then our work is likely to be of less value to other scientific efforts than if we can demonstrate that our results have proven themselves in the four participating experiments.

Stated in an over-simplified manner, the key question we need to analyze is this: how do the experiment's scientists process data? The information we need to capture includes:

- dataset types stored (both types of files and types/classes of object collections)
- definition of jobs and job types, what their control parameters are, and how they are expressed and invoked
- the grouping of jobs into processing pipelines that may be internally amenable to virtualization
- derivation dependencies for each dataset type (the sequence of transformations to create each dataset type)
- the frequency and priority of each data object, program, and information process
- with what mechanisms and frequency will jobs look up and locate data copies to use
- the location of the storage of each dataset type in a virtual organization (e.g., what tier / location the data resides at)
- the likely replication patterns of the datasets (where to, for how long, how/if replicas are eventually disposed of)

To begin with, it will be useful to create a high-level, abstract model of experiment data flow, to use as a guide to the data flow analysis. For example, it seems that at a very high level, most of our experiments follow a model that is something like:

Capture and/or simulate

Refine the raw or simulated data into a more manipulatable form

Re-factor and/or reorganize the data, sometimes changing the dimensions upon which it is based

Index the refined data, gradually building knowledge about the science phenomena inherent in the data

Distribute the data within the virtual organization for local processing

Analyze the data, typically through search, filtering, and statistical correlation techniques

Reprocess previous steps, backtracking as necessary, as algorithms, indexing, and search criteria are refined

We proposed to develop a common format for describing these processes, data flows and dependencies. We need to look for new ways to describe the changing rates of data production and consumption within each experiment, and to describe the manner in which data items depend on and are derived from each other. A good example of the beginnings of such an effort are represented in the documents that Koen Holtman has produced to describe the CMS experiment [references].

We see the analysis effort proceeding in 2 phases: Step 1 is to locate the best possible source information and make it broadly available to all GriPhyN by maintaining reference web pages that link to appropriate experiment activity pages. Step 2 is to analyze that information, and reduce it, and then extract the information from those sources into a common GriPhyN format.

Ideally, we will identify common information about each experiment in a common format using a common vocabulary. We need to determine the aspects of each experiment's data processing processes that are most relevant to GriPhyN's mission.

The following knowledge bases are proposed, to document the analysis. This information should be maintained as a continually updated GriPhyN document set:

- Data and Application Map – a chart showing data types, application tools, and their dependencies at a glance. By “data types” we mean: file types, object classes within persistent object-oriented databases, and tables within relational databases.
- Data dictionary – a detailed description of each data type, down to the level where computer scientists can see and understand the access patterns and the derivation dependencies of that data.
- Tool dictionary – a detailed description of each relevant application that will be part of Grid jobs and/or data “transformations”. We need to, in particular, understand these applications from a data transformation and data dependency point of view. What data objects are searched or read by the application, and what data objects are produced or transformed by the application? We also need to understand in detail how parameters and input/output specifications are passed to applications, and how some applications dynamically navigate around a massive information base.
- Data requirements spreadsheet – a summary of quantitative data storage and transfer requirements, detailing a time-varying birds-eye view over multi-year periods of how data will be produced, consumed, and replicated throughout the multiple sites and tiers of each experiment’s virtual organization. This data will be used primarily to determine the scaling needs for data transport and cataloging mechanisms, in terms of storage capacity, catalog capacity, and transaction rates.

Much of this knowledge base should consist of concise, reference-manual-style documentation that details specific data formats, tools, and science-driven IT processes. We want to collect and tabulate a useful reference base of information, rather than a lot of words. Where possible, this reference base should consist of pointers to the experiments’ document repositories.

An excellent start towards this analysis process can be seen in two documents from the GriPhyN-CMS collaboration:

CMS Data Grid System and Requirements (edited by Koen Holtman)

http://www.griphyn.org/documents/document_server/technical_report/2001/1/cmsreqs.pdf

Introduction to CMS from a CS viewpoint (written by Koen Holman)

http://www.griphyn.org/documents/document_server/technical_report/2001/4/koen_intro_cms.pdf

7.2 Challenge Problem Identification

We propose to conduct all integration of GriPhyN results into the experiments through the vehicle of *challenge problems*. This phrase is appropriate, in that we view this integration as the *most challenging* aspect of the entire GriPhyN program. Our partner experiments are large and complex: scientifically, technically, logistically, and organizationally. Challenge problems serve as a focal point of our efforts. They give the plan a concrete grounding, help identify integration points within experiments’ processes, and provide demonstrable results of clear value.

Challenge problem solutions involve integrating VDT components with application code and tools to yield working solutions that *are suitable for* live experiment usage. Examples of challenge problems and CP sequences that are created from the GriPhyN feature sets include:

- CP-1 Virtualize an application pipeline
- CP-2 High speed data transfer to replicate results
- CP-3 Automated planning
- CP-4 Mixed replication and re-materialization at high speeds
- CP-5 Abstract generator functions added to virtualization

CP-6 Jobs submitted from high-level tools/UIs (e.g., GRAPPA)

CP-7 Intelligent job management: Transparency, Fault Tolerance, Advanced policy and scheduling

CP-8 Monitoring and information synthesis

It is not easy for an orthogonal research program to insert its results into the mainstream of independent experiments, with independent schedules and, in many cases, funding and oversight. This is further complicated by the GriPhyN mandate to find common solutions across the experiments which runs counter to the need typically felt by each experiment for precisely tailored custom solutions to their complex software problems. The need for commonality is dictated in part by limited staff resources, but primarily by the need to produce results which can benefit numerous disciplines.

We will apply an “intercept” strategy to challenge problem design: we need to determine where the experiments will be when the GriPhyN results are expected to be ready for live usage; otherwise, the results will be irrelevant to the experiments. This will require that we identify integration points (both functionally and in the experiment’s schedule), negotiate the willingness of the experiments to accept and perform integrations, achieve timely deliverables, and track the experiments and their commitments to GriPhyN, so that we can adjust the GriPhyN plans to accommodate any changes that occur in the experiment’s plans.

In designing challenge problems, we need to clearly document the value proposition that the GriPhyN research results would bring to each experiment. In some cases, we will need to make a tradeoff between value to the experiment, difficulty of the challenge, and risk to the experiment for integrating a GriPhyN result. We need to be keenly aware of the quality assurance processes of the experiments if we are to propose integrating changes into mainstream tools upon which the experiments are critically dependent.

As part of the challenge problem identification, we need to develop a plan for how the solution to the problem will (or can) be ultimately integrated back into the experiments standard science processes.

8 Coordination Between Grid Projects

This section presents a high-level list of the anticipated relationships with external projects and organizations. Many of these are already in progress, and most have been discussed between several organizations. The details of these relationships will be planned and managed in many different documents and electronic forums.

PPDG

Expect from PPDG: GDMP; Scalability feedback and testing of distributed catalog solutions and high-end databases for catalogs; research into mechanisms for distributed physics analysis (Conrad Steenberg); research into fault tolerance (Takako Hickey)

Provides to PPDG: VDT, Data Grid Architecture document (for PPDG review and feedback); research results in such areas as scalable catalog solutions, data replication strategies, fault tolerance techniques.

Shares with PPDG: analysis of ATLAS and CMS data management requirements; development of US testbeds; application experiments.

iVDGL

Expect from iVDGL: acquisition and setup of hardware for testbeds; uniform installation of VDT; technology for operations of testbeds; feedback on VDT based on large-scale experiments; trace information for CS research into replication strategies, etc.

Provides to iVDGL: VDT; maybe components (beyond VDT) for construction of iGOC (open issue).

Shares with iVDGL: Hardening of VDT; conduct of experiments.

EUDG

Expect from EUDG: GDMP (via PPDG); Scalability feedback and testing of distributed catalog solutions and high-end databases for catalogs. Currently being discussed is whether we may receive (or co-develop) some additional components, e.g., schedulers.

Provides to EUDG: VDT.

Shares with EUDG: design and development effort for data grid components (RLS, RFT, RRT); conventions for toolkit installation; conduct of large-scale transatlantic experiments. We may jointly participate in Education and Outreach activities related to the EUDG-IE's work by Gianfranco Mascari, Emanuela Piervitali.

GRIDS Center

Expect from GRIDS Center: basic packaging of Globus and Condor to serve as base for VDT; improved hardening, installation, and documentation of these components; training; some support services to GriPhyN user community for VDT; outreach to additional user communities.

Provides to GRIDS Center: VDT.

Shares with GRIDS Center: Outreach to common user communities.

HICB/HIJTB

Expect from JTB: Coordination with international Data Grid communities.

Provides to JTB: Input on recommended technologies, testbed requirements and plans.

Shares with JTB: Development of standard technologies, development of testbeds, international experiments.

GGF

Expect from GGF: Best practices and standards in such areas as GridFTP, security.

Provides to GGF: Requirements analyses, best practices, and proposed standards.

Shares with GGF: Development of standards.

Globus Project™

Expect from Globus: All services of Globus 2.0 (GSI, MDS, GRAM, Replica catalog and management; GridFTP); advanced security technologies (CAS); RLS; RFT (to be developed with funding from DOE SciDAC and PPDG);

Provides to Globus: Requirements and requirement analyses; simulation results; monitoring tools

Shares with Globus: Requirements, techniques, and/or tools for eventual incorporation into and support by Globus, in such areas as monitoring and security.

DTF

Expect from DTF: Detailed specification of environment: Chips, compilers, OS, and data storage and transfer architecture; job execution and scheduling environment.

Provides to DTF: GriPhyN needs to arrange with experiments the porting and certification of some portion of their applications to the specific IA64 chip and platform needed to run the in

DTF. Potentially, an NMI-compatible version of the VDT to augment the base Grid software on DTF nodes.

Shares with DTF: Outreach to user communities, development of requirements for testbeds and tools.

8.1 Coordination Regarding Virtual Data Toolkit

This work will be done in collaboration with the GRIDS Center and the NCSA Alliance, and will integrate software from other DOE-and NSF-funded Globus and Condor development activities. We envision this work providing the base software installation for the iVDGL.

The GriPhyN contributions to this work will be as follows:

- Addition of various components, over time (e.g., in the first year, DAGMan)
- Definition of a standard software release, consistent with the Data Grid Reference Architecture (DGRA).
- In collaboration with the GRIDS Center, packaging, integration, and documentation to produce an easily installable, documented binary version of this software release.
- In collaboration with the GRIDS Center, provide support for the software.

We assume that the iVDGL will handle:

- Operation of central services for GriPhyN experiments that wish to use this.
- Deployment of software on resources to construct testbeds.
- Development of push mechanism for automatic distribution of updates.

9 Project Logistics

In this section we described project plans for Reporting, Communications, and Meetings.

9.1 Coordination Meetings

We hold the following types of meetings:

- All-hands meeting: twice a year. Plans are presented here, people report on progress, etc.
- Periodic applications-software meetings to discuss software development and iVDGL deployment: twice a year (half-way in-between the two all-hands meetings). These provide input to the next round of planning.
- Periodic application-CS one-on-one meetings: structured as an “all-hands” on each side. Ideally these are held twice a year, also.
- Occasional CS research meetings: as needed, on specific topics.

9.2 Communications

We maintain:

- Various email lists, all archived.
- A web site, www.griphyn.org, with news etc., a calendar of upcoming events, and an archive of material from past events.

- Analysis and design documents for each experiment, and for elements of the VDT architecture.
- The VDT and associated documents.
- A Data Grid Pubs web site with relevant publications.
- Status reports, collated and made available on the GriPhyN web site.

9.3 Planning

We maintain a set of GriPhyN Project Plan documents, revised on a six-monthly basis prior to each all-hands meeting, and presented at the all-hands meeting. The documents consist of this master plan document, plus one yearly planning document per experiment. Eventually we will probably break the CS Research, VDT, and Testbed activities into standalone planning documents.

We also plan to integrate the use of a planning tool (such as MS Project) into our planning process to track detailed task and milestone lists.

9.4 Reporting

We provide NSF with a yearly report once each year, in June. The June 2001 report is posted as:

GriPhyN Annual Report for 2000-2001, NSF Grant 0086044

http://www.griphyn.org/documents/document_server/status_report/2001/1/griphyn_report2001_final.pdf

9.5 Project Personnel

Note: The following table is now out of date and will be revised shortly. It still, however, correctly reflects the approximate division of resources across the project's institutions.

Table 1 shows the allocation of graduate students, postdocs, and staff to the various elements of the project. In the initial budget allocation, the institutions with larger allocations have some ramp up in the first year; some CS institutions do not receive funding for graduate students in the fifth year. We emphasize that while we show a scenario for approximately constant level of effort at each institution, we do not expect that we will stick to a static budget for a five-year project. The management structure outlined in the proposal will work to optimize the use of resources during the project period.

We anticipate funded personnel being mapped to activities roughly as follows:

- CS research will involve 9 graduate students, and 4 postdoctoral associates.
- Virtual Data Grid Toolkit development will involve 3 graduate students, and 4 technical staff.
- Discipline applications will involve 2 graduate students and 6 postdocs.
- Outreach and education involves one dedicated staff member.

Table 1: Personnel per project and institution, categorized as Graduate students, Postdoc, or Staff.

Inst.	CS Research			VDT			Physics Experiments											
							ATLAS			CMS			LIGO			SDSS		
	G	P	S	G	P	S	G	P	S	G	P	S	G	P	S	G	P	S
Caltech											1	0.5			0.5			
IU							2	1										
JHU																1	1	
NWU		1																
UC	1	1	*	1		2*												
UCB ¹	3		*															
UCSD	2		*															
UF ²		1									1							
USC		1	*	1														
UTB														1 ³				
UW-Mad	3		*	1		2*												
UW-Mil															1 ⁴			
Total	9	4		3		4	2	1			2	0.5		1	1.5	1	1	

This is based on the original proposal; we need to update to reflect final budgets. Note that some of the larger budgets were somewhat reduced in the first year. A * denotes summer salary for faculty.

9.6 Faculty

The following is a list of the faculty involved in the project:

- CS research: Arpacci-Duseau, Foster, Franklin, Kesselman, Livny, Marzullo, Moore, Otoo, Schopf, Taylor
- VDT development: Foster, Kesselman, Livny
- ATLAS: Gardner, Huth
- CMS: Avery, Bunn, Newman
- LIGO: Allen, Lazzarini, Campanelli, Romano
- SDSS: Szalay

¹ UCB graduate students are only funded for 4 years.

² UF also has cost sharing for a project coordinator and part of an administrative assistant

³ The UT Brownsville LIGO staff person is Manuela Campenella, who is in fact focused primarily on Education and Outreach.

⁴ The UW Milwaukee staff member, Scott Koranda, is funded 1/3 by NCSA in support of GriPhyN.

9.7 Project Team Structure

We structure our research project as three distinct but tightly integrated activities:

- **IT research:** groundbreaking IT research motivated and guided by challenges encountered by domain scientists in meeting their computational and data management needs.
- **Application experiments:** prototyping new information technology and interfacing it with scientific applications in real-life test-bed environments defined by CMS, ATLAS, LIGO, and SDSS requirements.
- **Tool development:** turn “winning” prototypes into production quality tools to be used by the scientific community.

The 26 researchers—12 computer scientists and 14 experimental physicists—that we have brought together to attack these problems have been intrigued by the research challenges, the possible impact, and the scope of this project. These researchers represent a wide inter- *and* intra-disciplinary spectrum of research interests, talent, and experience. The computer scientists are organized in seven groups—Berkeley, Chicago, Indiana, NWU, San Diego, USC, and UW Madison—each managed by local GriPhyN personnel and each contributing a unique capability to the project and responsible for one major research activity. The physicists are distributed across a comparable number of institutions, chosen with a view to IT expertise and connections with the physics experiments.

The rich collection of technology and software already developed by project participants means that we can start all three of the phases just listed in parallel. The result will be a steady stream of IT research results, application experiences, and production quality software.

We plan to coordinate these diverse efforts via the definition and frequent review of a *PVDG architecture* that defines the interfaces between key components, and the scheduling of frequent *integration events* in which components developed by different groups are brought together for interoperability testing.”

10 Education and Outreach

The Education and Outreach (E/O) program of the GriPhyN project is designed primarily to expose faculty and students at other U.S. universities and institutions to GriPhyN research. In particular, it intends to promote learning and inclusion via the integration of a diverse set of minority and under-represented institutions into the scientific program of participating physics and computer science experiments. This program will engage all GriPhyN senior personnel, with each committing to lecturing and mentoring activities at these institutions.

In order to facilitate and coordinate these activities, The University of Texas at Brownsville recently hired a full-time faculty member, Manuela Campanelli, to serve as E/O coordinator for GriPhyN, with a start date of Fall 2001. Our plans for E/O are described below.

10.1 Web page for GriPhyN E/O

The E/O web page will grow to contain basic educational material about data grids and the participating physics experiments. In addition, it will provide basic technical support information (e.g., documentation, user manuals, how-to guides, etc.) for the GriPhyN virtual data toolkits. We also plan to extend a web-site that Alex Szalay (at Johns Hopkins University) and Jim Gray (at Microsoft) are developing for accessing SDSS data, to illustrate the concept of virtual data. One

idea is to produce on-the-fly custom data sets representing images of the sky as viewed in different frequency bands.

The evolving GriPhyN Education and Outreach Web page is available now, at

<http://www.aei-potsdam.mpg.de/~manuela/GridWeb/main.html>

and is linked to from the main GriPhyN web page at www.griphyn.org.

10.2 Research Experience for Undergraduates (REU) supplement

In Fall 2001, Campanelli plans to submit a proposal to NSF requesting an REU supplement to support students doing grid-related research. If funded, students would be able to work on research projects during the summer months at participating GriPhyN (or iVDGL) institutions. At the end of the year, the students would present posters or give talks at a conference specifically designed to showcase their work. (Note: Philip Dukes, another faculty member at UT Brownsville, who has experience in physics education, may serve as co-PI on this proposal.)

10.3 Grid-enable the UT Brownsville Linux cluster

UT Brownsville has nearly completed constructing a 96-node Linux-cluster. Although the cluster will be used primarily for LIGO data analysis, it can also serve as a testbed for GriPhyN software and be used to introduce minority students at UT Brownsville to distributed computing and grid-related technology. If the iVDGL proposal and the REU supplement get funded, we will be able to support an additional two or three undergraduate students to learn how to install and run grid software, like Globus or Condor, on the completed cluster.

10.4 Involving other minority serving institutions

If the iVDGL proposal is funded, two other minority-serving institutions (Hampton University and Salish Kootenai College) will receive funds for hardware and personnel to construct small clusters (i.e., Tier3 centers), thus bringing a large number of additional minority students directly into contact with large-scale grid research. Campanelli will coordinate the E/O effort for the iVDGL project as well, providing technical support for the Tier3 centers and functioning as an interface with the GriPhyN research community. Moreover, Keith Baker, who would be the lead person for the Hampton University Tier3 center, is a principal investigator for QuarkNet.

10.5 Leveraging on-going existing E/O programs

There are a number of on-going education and outreach programs that we plan to utilize in the forthcoming months:

- Indiana University and The University of Florida are participating GriPhyN institutions and active QuarkNet centers. GriPhyN researchers at these institutions will be encouraged to provide a grid-related component to the already existing QuarkNet activities.
- Valerie Taylor (a GriPhyN senior investigator at Northwestern University) is a PI for the Coalition to Diversify Computing project within EOT-PACI---the education, outreach, and training partnership for advanced computing resources. Campanelli plans to work with Taylor on ways to link the E/O activities of GriPhyN with the EOT-PACI program.
- We have already begun talks with ThinkQuest to develop special challenge projects based on the application sciences and grid technology. GriPhyN (or the iVDGL project) would provide resources in the form of interesting data sets (e.g., SDSS images or LIGO data) and/or "sandbox" CPUs that students could use when creating innovative web-based educational tools for ThinkQuest competitions.

10.6 Course development

All GriPhyN senior investigators will be encouraged to include grid concepts in their physics or IT courses. In addition, each GriPhyN senior investigator is committed to lecturing and mentoring activities at other (in particular minority serving) institutions. These activities are important as they are specifically targeted to promote and improve the education of students regarding grid-related activities.

10.7 Workshops and tutorials

Campanelli, together with several other GriPhyN senior investigators, will organize several 'how to' workshops and tutorials at the various Tier3 minority serving institutions. In order to facilitate the participation of all senior investigators and given the limited E/O budget, we propose to do this in conjunction with at least one of the all-hands GriPhyN meetings. This would allow a more direct and larger participation of minority students, without needing additional travel money for students. UT Brownsville has volunteered to hold such a meeting in Spring 2003.

10.8 Other activities

An interesting possibility for GriPhyN E/O is the creation of an educational documentary about GriPhyN computational data grids. Because this project would require significant budgetary and human resources that are not presently available in our E/O program, we are considering the possibility of working together with other national or international grid partners, like the European DataGrid, which has expressed an explicit interest in this direction. Such a documentary would certainly produce a worldwide impact on the general public, and could also be used as a powerful media to reach high-school students and teachers.

Additional Specific Education/Outreach (E/O) goals for Oct. 2001 – Oct. 2002:

1. Since the E/O program of the GriPhyN and iVDGL projects is designed to expose faculty and students at other U.S. universities and institutions to grid-related research, it is extremely important that all GriPhyN and iVDGL senior personnel be committed to lecturing and mentoring activities at other institutions. We will keep a record and make available a list with materials of talks given about the grid and GriPhyN.
2. Our second near term goal is to give undergraduate students the opportunity to participate in grid-related research at GriPhyN/iVDGL institutions by taking advantage of the NSF Research Experience for Undergraduate (REU) Program. We will submit a proposal for an REU supplement early in 2002.

We will seek funds to support 10 to 20 undergraduate students to do GriPhyN related research at various GriPhyN institutions. Students themselves will then apply and choose the mentoring institutions on the basis of the research projects proposed by each institution. This will already provide some pre-selection process. Further selection will be done by the mentor based on the individual skills required for the chosen project.



Technical Report GriPhyN-2001-xxx

www.griphyn.org

GriPhyN Research for ATLAS

Year 2 Project Plan

ATLAS Application Group

GriPhyN Collaboration

&

Software and Computing Project

U.S. ATLAS Collaboration

12/21/2001

Version 3.7

Table of Contents

1	Introduction	4
1.1	High Level Goals	5
1.2	Terminology and Acronyms	6
2	ATLAS	7
2.1	ATLAS Software Overview	7
2.2	ATLAS Data Challenges	8
2.3	Personnel	10
3	Manager of Grid-based Data – Magda	11
3.1	Magda references:	13
3.2	Magda Schedule	13
4	Grid Enabled Data Access from Athena – Adagio	14
4.1	Adagio Schedule	15
5	Grid User Interface – Grappa	15
5.1	Grid Portals	15
5.2	Grappa Requirements	16
5.2.1	Use Cases	16
5.3	Grappa and Existing Tools	16
5.3.1	XCAT Science Portal	16
5.4	XCAT Design Changes for Grappa	17
5.5	Grappa and Virtual Data	18
5.6	Grappa Schedule	19
6	Performance Monitoring and Analysis	19
6.1	Grid-level Monitoring	20
6.2	Local Resource Monitoring	22
6.3	Application Monitoring	23
6.4	Performance Models	24
6.5	Grid Telemetry	24
6.5.1	Prototype Grid Telemetry System	25
6.5.2	Telemetry Program of Work	26
6.6	Monitoring Schedule	27
7	Grid Package Management – Pacman	28
7.1	Pacman Schedule	30
8	Security and Accounting Issues	31
9	Site Management Software	31
10	Testbed Development	32
10.1	U.S. ATLAS Testbed	32
10.2	iVDGL	33
10.3	Infrastructure Development and Deployment	33
10.4	Testbed Schedule	34
11	ATLAS – GriPhyN Outreach Activities	34
11.1	Outreach Schedule	35

12	Summary, Challenge Problems, Demonstrations	36
12.1	ATLAS Year 2 (October 01 – September 02)	36
12.1.1	Goals Summary	36
12.1.2	Challenge Problem I: DC Data Analysis	36
12.1.3	Challenge Problem II: Athena Virtual Data Demonstration	37
12.1.4	Demonstrations for ATLAS Software Weeks	38
12.1.5	Demonstration for SC2002	38
12.2	ATLAS Year 3 (October 02 – September 03)	39
12.2.1	Goals Summary	39
12.2.2	Challenge Problem III: Grid Based Data Challenge	39
12.2.3	Challenge Problem IV: Virtual Data Tracking and Recreation	39
12.3	Overview of Major Grid Goals	40
13	Project Management	40
13.1	Liaison	41
13.2	Project Reporting	41
14	References	41

1 Introduction

The goal of this document is to provide a detailed GriPhyN – ATLAS plan for Year 2. Some high-level plans for Year 3 are also included.

As an application, ATLAS software is inherently fit for Grid computing with its distributed data and computing needs. As such, we have developed a coordinated approach for the various U.S. Grid projects, namely PPDG, GriPhyN and iVDGL, focused on delivering the necessary tools for the ATLAS Data Challenges (DCs).

These tools fall into several different categories. First, and mainly as part of PPDG although it will be used for GriPhyN, is Magda – the Manager for Grid-based Data. This tool is a data distribution tool/sandbox that is being used for initial work in distributed data management. It was developed to enable the rapid development of components to support users, and as other project pieces reach maturity they are easily incorporated. For example, GridFTP was recently incorporated, and current testing using the Globus replica management tools is underway, replacing original prototype code for both functions. This is detailed in Section 3. Related to this work is Adagio (Athena Data Access using Grid I/O) to enable data-access over the Grid from within the Athena infrastructure. This is described in Section 4.

Second is Grappa, the Grid Portal for Physics Applications, which is being developed as a user-friendlier interface to job submission and monitoring. It will interface to Magda in future development as well. This work is based on the Indiana XCAT Science Portals project, and is fully adaptable to new developments in Grid software and in the ATLAS/Athena framework itself. Initial work has focused on a simpler, web-based interface to job submission, with easy access to Python job scripting. This is fully described in Section 5.

In the area of monitoring, there are several on-going efforts with respect to different aspects of the problem. We are leading the joint PPDG/GriPhyN effort in monitoring to define the use cases and requirements for a cross-experiment testbed. In addition, we have been evaluating and installing sensors to capture the needed data for our testbed facilities internally, and determining what information should be shared at the Grid level, and the best ways to do this. At the application level, much work has been done with Athena Auditor services to evaluate application performance on the fly. For visualizing monitoring data, we have developed GridView to easily see the resource availability on the U.S. ATLAS Testbed. These efforts are detailed in Section 6.

We have also been working on the development of many needed management tools. Section 7 discusses Pacman, our package management system, which is a candidate for the packaging of the VDT. Section 8 addresses our security approaches, or rather, our use of other people's work in security, and Section 9 describes our approach to site management software. Section 10 describes testbed development. Section 11 describes education and outreach activities. Section 12 summarizes GriPhyN – ATLAS goals, and Section 13 provides information about project management.

1.1 High Level Goals

The high level goals for Year 2 of GriPhyN – ATLAS are

1. Provide support for analysis of ATLAS Data Challenge 1 data collections. The reasons for this approach are:
 - To drive development of GriPhyN technologies in several key areas including Grid data access and management, grid user interfaces, monitoring and security.
 - To demonstrate added value by GriPhyN to immediate ATLAS project objectives.
 - To acquire support within ATLAS for adoption of GriPhyN technologies, important for future VDT releases which will emphasize virtual data methods.
 - Coordinate GriPhyN virtual data research with other U.S. Grid projects including PPDG and iVDGL.
 - To forge GriPhyN ties with the ATLAS Data Challenge team in which participants from several Grid development teams and testbed efforts centered at CERN and the EU work to support ATLAS computing objectives.
2. Demonstrate large scale instantiation of compute resources comprising the hierarchy of LHC Computing Model Tiers 0-3 through design, development and testing of site management and software packaging tools.
3. Create ATLAS demonstrations for SC and other venues (such as CHEP) which exhibit cradle-to-grave Grid-level analysis of ATLAS high energy physics data, operative from the point of view of a physicist-user at the Tier 3 level.
4. Explore and/or design ATLAS instances of virtual data tracking and catalog architectures being developed by the GriPhyN – CMS and LIGO application teams leading to specification and development of virtual data toolkit components for ATLAS in Years 3-5.

Technical goals for Year 2 of GriPhyN – ATLAS are:

1. Provide easy access mechanisms to DC1 data using Magda, enhancements to Magda which capture metadata attributes created during DC1 production, and other collection navigational tools.
2. Support Grid file replication and data distribution efforts (GridFTP) for distribution of DC1 data from production sites (CERN and a few Tier 1 sites) and the Tier 1 at Brookhaven, and the two ATLAS prototype Tier 2 sites (part of iVDGL) at Boston University and Indiana University.
3. Register DC1 data caches at these sites with Magda.
4. Continue development of Pacman source distribution caches for VDT, ATLAS, and other external software packages as required.
5. Continue development of ATLAS remote site execution environment and startup kits.
6. Develop simple Grid job submission tools based on the Grappa portal.
7. Deploy a grid information service for ATLAS / iVDGL sites based on the MDS2 service.
8. Deploy an ATLAS software information service which describes ATLAS software

- installations at Grid sites based on MDS.
9. Connect Grappa with grid information service describing Grid resources available to ATLAS users.
 10. Demonstration series of various pieces of the ATLAS production and analysis chain for Monte Carlo data
 11. Demonstration of Grid-based data analysis using ATLAS software at a significant number of Grid sites, beginning first with Tiers 0-2, later expanding to ATLAS Tier 3 sites, and later to non-ATLAS sites such as sites within iVDGL home to the other GriPhyN application teams.
 12. Demonstrate connectivity of Grid-based data analysis jobs based on GriPhyN technology to DataGrid Testbed sites.

1.2 Terminology and Acronyms

Several acronyms are used within this document. They include the following project related acronyms:

DC	Data Challenge, defined by the ATLAS project
GG	GriPhyN – ATLAS goals as defined in this document
GM	GriPhyN – ATLAS milestone as defined in this document
iVDGL	International Virtual Data Grid Laboratory Project
VDT	Virtual Data Toolkit, developed by GriPhyN and supported by iVDGL
PG	PPDG – ATLAS project goals, as defined by PPDG project plans
PPDG	Particle Physics Data Grid Collaboratory Project
EU DG	European DataGrid project

In addition, in sections below we identify work items, approximate schedules, and significant milestones to mark progress. Where appropriate, we cross reference this to the Grid planning schedule for U.S. ATLAS which include activities from other Grid projects such as PPDG, iVDGL, liaison and integration tasks associated with the EU DataGrid testbed effort, the HENP Networking Working Group, etc. Below is an example, with work area key following.

Table 1 Example work item list and milestones

GriPhyN Code	ATLAS Grid WBS	Name	Description	Date Start	Date End
Type-X1	1.1.x	Short name for project from work area X	More detail	Year-Quarter	Year-Quarter
Type-X2	1.2.x				
Milestones					
Type-X1	1.1.x	Short name for project milestone from work area X	More detail	Date	

Table 2 Keys denoting work areas within GriPhyN - ATLAS

Type		Project area X	
GG	GriPhyN Goal	D	Grid Data Access from Athena
GM	GriPhyN Milestone	DM	Data management, Magda
CP	Challenge Problem	P	Packaging
GD	GriPhyN Demonstration	I	Interface, Grappa
		T	Testbed
		M	Monitoring
		0	Education and Outreach

2 ATLAS¹

This section gives a basic overview to the ATLAS software environment, describes the data challenges which drive the computing goals for ATLAS, and lists the involved personnel.

2.1 ATLAS Software Overview

Athena² is the common object oriented framework used by the ATLAS experiment for simulation, reconstruction, event filtering, and analysis. It is based on the GAUDI³ architecture developed by the CERN LHCb collaboration. Development of the GAUDI kernel has since become a joint, multi-experiment project as other HEP experiments have since adopted the framework. ATLAS software is still in a migratory phase from previous Fortran-based procedural codes, such as ATLSIM, ATRECON, etc, and the Fortran based HEP event simulation package GEANT-3, to the new OO framework. The state of affairs is the result of tactical decisions, made at the international ATLAS level, to use well understood, benchmarked codes for the extensive physics and detector performance studies which formed the basis of the *Detector and Physics Performance Technical Design Report*⁴. The decision resulted in the successful validation of the ATLAS spectrometer design, but a necessary consequence of the approach was to delay the transition to the new OO-based framework. As such, the core ATLAS software is today in a highly developmental phase, and so in some cases Fortran legacy codes are used for preliminary Grid toolkit evaluations.

The Athena architecture, indicated by the object diagram of Figure 2-1, supports multiple data persistency services and insulates user code from the underlying storage technology. Physicists supply algorithms which perform tasks such as track finding and fitting, vertex finding, cluster finding and reconstruction. Users interact with the Athena through use of job options files, and in the near future, by a Python scripting interface. Adagio, discussed in Section 4 below, is an effort within PPDG and the core ATLAS database groups to examine the connectivity layer between Grid and core ATLAS persistency services. The run time environment is complex, as user algorithms are dynamically linked with shared object libraries, resident on the local machine or accessible from a remote site via AFS. Other files, such as parameter files and conditions databases, need to be setup and configured properly.

A major goal of ATLAS-GriPhyN is to create the Grid interfaces for collection of Athena services (such as EventSelector, histogram, auditors, messaging and monitoring services), and to

provide research tools which can be more broadly useful within the GriPhyN collaboration. For example, Pacman is a software management tool to aid in the deployment of software packages, helping maintain consistency among software distributed across multiple administrative domains on the Grid. Magda is a data management tool used for viewing, accessing, and adding to Grid-distributed data with interfaces to C, Java, Perl, and the Web. Grappa is a high-level user interface based on science portal technology, allowing physicists to launch jobs, monitor them, and interact with Grid data tools without having to learn the details of Grid programming. Such a high-level interface is more than a GUI for ATLAS: GriPhyN research entails multiple approaches for problems such as metadata management, and Grappa is designed with a software component plug-and-play architecture that allows using any or all of those different approaches.

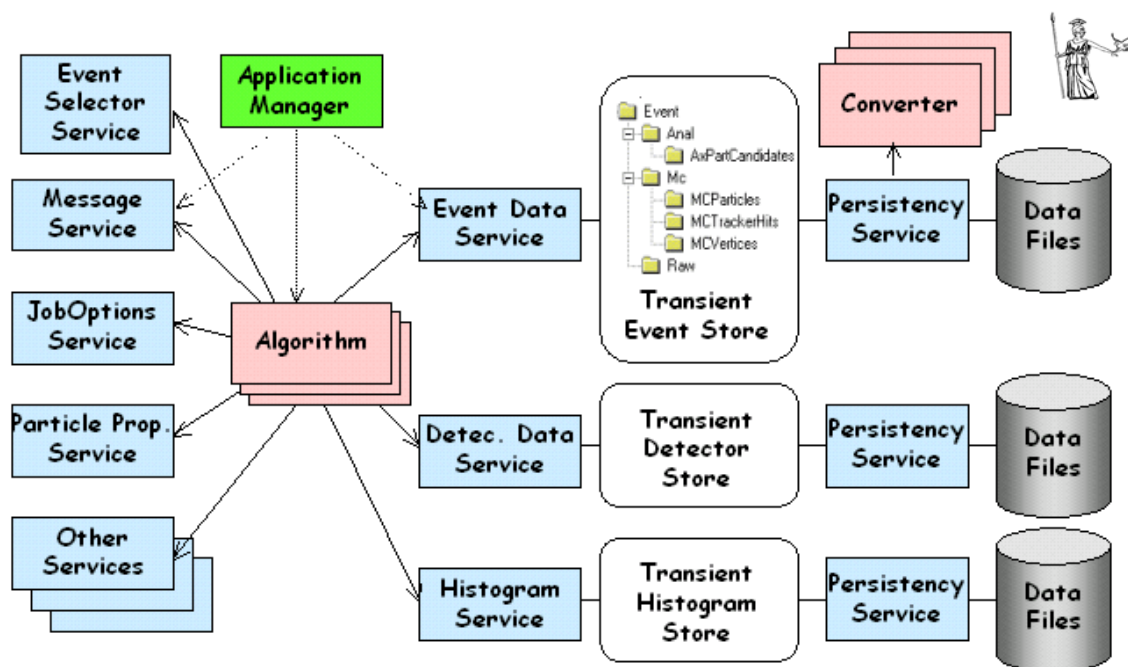


Figure 2-1 Athena Object Diagram

2.2 ATLAS Data Challenges

The ATLAS collaboration will undertake a series of data challenges* in order to validate the LHC Computing Model, which underwent an extensive CERN review concluding in January 2001. The validation will address all aspects of ATLAS software, especially its control framework and data model, and eventually choices in Grid technology. The data challenges will be executed at prototype Tier computing centers, and will be of increasing complexity and scale to exercise as much as possible Grid middleware technologies. The results of the Data

* See http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/DC/dc_page.html for more information about the DC effort within ATLAS.

Challenges will be used as input for a Computing Technical Design Report due by the end of 2003 for ATLAS.

The first three data challenges (DC) that will be run starting in December of 2001, and will last until December of 2003. The DCs are designed to have physics content in order to draw a large group of data analysts from the physics community, thus providing a better check and exercise of the software. Data Challenge 0 (DC0), which runs in December 2001 and January 2002, is essentially a test of the continuity of the code chain. DC0 will provide ATLAS with continuity tests of several key data paths:

Generators --> full simulation --> reconstruction --> analysis;
Generators --> fast simulation --> analysis;
Physics TDR[†] data --> reconstruction --> analysis.

A by-product of these continuity tests will be a reference set of scripts and related job options files used to validate each link in the three chains. Each of these recipes essentially defines a transformation of one data product into another, and each of these "standard" recipes will be of interest to the ATLAS collaboration at large. These standard job options and scripts will provide a foundation for our prototyping of transparency with respect to materialization, and serve as a basis for our initial transformation catalog. Only modest DC0 samples will be generated, and essentially all in flat "traditional" sequential file format. DC0 production will likely take place at CERN only, though Tier 1 sites will be testing their software environments with DC0 executions in preparation for DC1.

Data Challenge 1 (DC1) will run during the first half of 2002, and will be divided in two phases. In the first phase several sets of 10^7 events for high-level trigger studies will be generated. The second phase will be oriented to physics analysis, with several types of sets generated, some as large as 10^7 events. The second phase will also focus on testing new software, including Geant4, the new event data model and the evaluation of database technologies such as Root-I/O. The production of DC1 data will involve CERN and also sites outside of CERN, such as the Brookhaven Tier 1 facility. Several hundred PCs world-wide will participate.

Data Challenge 2 (DC2) runs for the first half of 2003. The scope will depend on what was accomplished in DC1, but the main goal will be to have the full new software in place. We will generate several samples of 10^8 events, mainly in OO-databases, and with large-scale physics analysis using Grid tools.

All the Data Challenges will be run on Linux systems operating according to ATLAS specifications, and with the compilers distributed with the code if not already installed locally in the correct versions. The DCs are summarized in Table 3 below. Subsequent to the planned DC0-DC2 challenges will be Full Chain and 20% scale processor farm tests. The detailed plans for these challenges will depend on the results of the first three DCs.

Our approach within GriPhyN is to design goals and milestones in coordination with, and in support of, the major software and computing activities of the international ATLAS Collaboration. Hence the attention paid to these DCs. This work is being done in close conjunction with specific Grid planning⁵ underway within the U.S. ATLAS Software and

[†] TDR = technical design report for detector and physics performance, as previously mentioned

Computing Project, which includes planning for the Particle Physics Data Grid Project (PPDG) and iVDGL.

Table 3 Schedule and Specifications for ATLAS Data Challenges

Name	Date	Events #, size	CPU SI95-sec	Data Volume	Description
DC0	December 01 to February 02	10^5 2.5 MB	10^8	1 TB	Continuity check of ATLAS software
DC1	February 01 to July 02	10^7 2.5 MB (larger if higher luminosity or if hits and digits written out)	Simulation: 3×10^{10} Recon: 6×10^9	Simulation: 20 TB Reconstruction: 5 TB (Multiples of this if pileup is assumed and hits are written out.)	Major test of production capabilities; 1% scale relative to final system. Grid tools to be used in analysis phase.
DC2	January 03 to September 03	10^8	10^{12}	100 TB, but perhaps as much as 50% of the full scale	10% scale test. Large scale production deployment of multi-tiered distributed computing services.
Full Chain Test	July 04	10^8	10^{12}	TBD	Test of full processing bandwidth, from high level trigger through analysis. High throughput testing of distributed services.
20% Processing Farm Prototype	December 04	10^9	10^{13}	Up to 0.5 PB	Production processing test with 100% complexity (processor count), 20% capacity system relative to 2007 level. High throughput, high complexity testing of distributed services.

2.3 Personnel

The ATLAS – GriPhyN team, Table 4 , involves participation from a number of individuals from ATLAS affiliated institutions and from computer scientists from GriPhyN university and laboratory groups. In addition, there is significant joint participation with PPDG funded efforts

at ANL and BNL.

Table 4 ATLAS – GriPhyN Application Group

Name	Institution	Affiliations	Role	Work Area
Rich Baker	BNL	PPDG, ATLAS	Physicist	Testbed, monitoring
Randall Bramley	IU	GriPhyN	Computer Scientist	Grappa
Kaushik De	UTA	ATLAS	Physicist	GridView, Testbed
Daniel Engh (start 2/02)	IU	GriPhyN, ATLAS	Physicist	Athena – Grappa, grid data access
Lisa Ensman (till 4/02)	IU	GriPhyN, ATLAS	Physicist	Athena – Grappa
Dennis Gannon	IU	GriPhyN	Computer Scientist	Grappa
Rob Gardner	IU	GriPhyN, ATLAS	Physicist	Project lead, physics contact
John Huth	HU	GriPhyN, ATLAS	Physicist	Management
Fred Luehring	IU	ATLAS	Physicist	ATLAS applications
David Malon	ANL	PPDG, ATLAS	Computer Scientist	Athena Data Access
Ed May	ANL	PPDG, ATLAS	Physicist	Testbed coordination
Jennifer Schopf	ANL	GriPhyN, Globus, PPDG	Computer Scientist	CS contact, Monitoring
Jim Shank	BU	GriPhyN, ATLAS	Physicist	ATLAS applications
Shava Smallen	IU	GriPhyN	Computer Scientist	Grappa
Jason Smith	BNL	ACF, ATLAS	Physicist	Monitoring, Testbed
Valerie Taylor	NU	GriPhyN	Computer Scientist	Athena Monitoring
Alex Undrus	BNL	ATLAS	Physicist	Software Librarian
Torre Wenaus	BNL	PPDG, ATLAS	Physicist	Magda
Saul Youssef	BU	GriPhyN, ATLAS	Physicist	Pacman, ATLAS app.
Dantong Yu	BU	PPDG, ATLAS	Computer Scientist	Monitoring

3 Manager of Grid-based Data – Magda

Magda (MANager for Grid-based Data) is a distributed data manager prototype for Grid-resident data. Magda is being developed by the Particle Physics Data Grid as an ATLAS/Globus project to fulfill the principal ATLAS PPDG deliverable of a production distributed data management system deployed to users and serving BNL, CERN, and many U.S. ATLAS Grid testbed sites (currently ANL, LBNL, Boston University and Indiana University). The architecture is illustrated in Figure 3-1. The objective is a multi-point U.S. Grid (in addition to the CERN link)

providing distributed data services to users as early as possible. Magda provides a component-based rapid prototyping development and deployment infrastructure designed to promote quick in-house development of interim components later replaced by robust and scalable Grid Toolkit components as they mature.

These work statements refer to components of U.S. ATLAS Grid WBS 1.3.3.3 (Wide area distributed replica management and caching) and WBS 1.3.5.5 (Infrastructure metadata management).

The deployed service will be a vertically integrated suite of tools extending from a number of Grid toolkit components (listed below) at the foundation, through a metadata cataloging and distributed data infrastructure that is partly an ATLAS-specific infrastructure layer and partly a generic testbed for exploring distributed data management technologies and approaches, to primarily experiment-specific interfaces to ATLAS users and software.

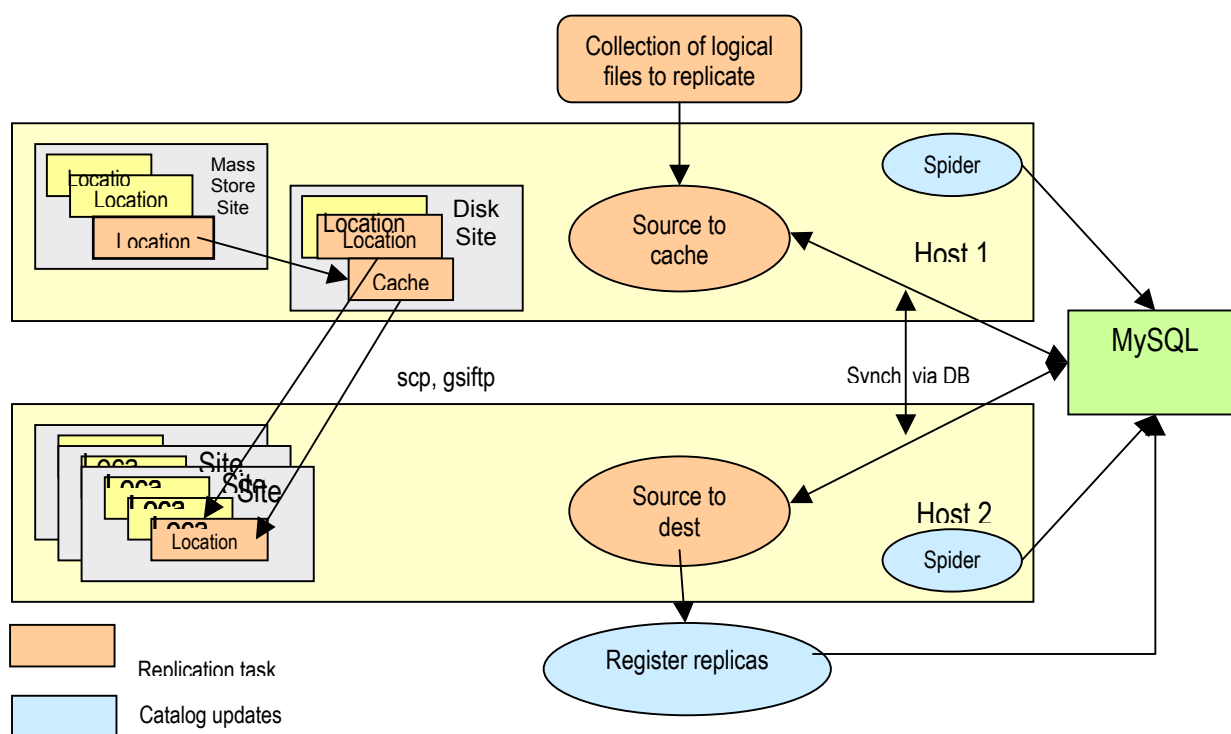


Figure 3-1 Magda Architecture (PPDG)

Grid Toolkit tools in use or being integrated within Magda include Globus GridFTP file transfer, GDMP replication services, Globus replica catalog, Globus remote execution tools, and Globus replica management.

Magda has been in stable operation as a file catalog for CERN and BNL resident ATLAS data since May 2001 and has been in use as an automated file replication tool between CERN and BNL mass stores and U.S. ATLAS Grid testbed sites (ANL, LBNL, Boston, Indiana) since

summer 2001. Catalog content fluctuates but is typically a few 100K files representing more than 2TB of data. It has been used without problems with up to 1.5M files. It will be used in the forthcoming ATLAS Data Challenges DC0 (Dec 2001-Feb 2002) and DC1 (mid to late 2002). In DC1 a Magda version integrated with the GDMP publish/subscribe data mirroring package (under development within PPDG and EUDG WP2) will be deployed. The principal PPDG milestone for Magda is fully functional deployment to general users as a production distributed data management tool in June 2002. The principal GriPhyN/iVDGL milestone is Magda-based delivery of DC1 reconstruction and analysis data to general users throughout the U.S. ATLAS Grid testbed within 2 months following the completion of DC1.

In addition to its role in early deployment of a distributed data manager, Magda will also serve as a development tool and testbed for longer term R&D in data signatures (dataset and object histories comprehensive enough to permit on-demand regeneration of data, as required in a virtual data implementation) and object level cataloging and access. This development work will be done in close collaboration with GriPhyN/iVDGL, with a GriPhyN/iVDGL milestone to deliver dataset regeneration capability in September 2003.

In mid 2002 Magda development in PPDG will give way to an emphasis on developing a distributed job management system (the PPDG ATLAS Year 2 principal deliverable) following a similar approach, and building on existing Grid tools (Condor, DAGman, MOP, etc.). This work will be done in close collaboration with GriPhyN/iVDGL development and deployment work in distributed job management and scheduling.

ATLAS GriPhyN/iVDGL developers plan to integrate support for Magda based data access into the Grappa Grid portal now under development (see Section 5).

3.1 Magda references:

Magda main page: <http://ATLASsw1.phy.bnl.gov/magda/dyShowMain.pl>

Magda information page: <http://ATLASsw1.phy.bnl.gov/magda/info>

PPDG BNL page: <http://www.usATLAS.bnl.gov/computing/ppdg-bnl/>

3.2 Magda Schedule

Table 5 Magda work items and milestones as related to GriPhyN

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-DM1	1.3.3	Deployment of basic services	Setup of Magda infrastructure for use in DC1 at GriPhyN / iVDGL sites; use of Pacman	Y2-Q1	Y2-Q4
GG-DM2	1.3.5	Metadata development	Magda metadata management	Y2-Q2	Y2-Q4

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
			interfaces for GriPhyN		
GG-DM3	1.3.3	Job submission interfaces	Interface to Grappa portal and other grid user interfaces	Y2-Q3	Y3-Q1
GG-DM4	1.3.3	Virtual data extensions	Development of Virtual Data signature tools	Y2-Q4	Y3-Q4
Milestones					
GM-DM1	1.3.3	Magda population	Construction of Magda database cataloging DC1 data will occur automatically during DC1 production.	8/1/02	
GM-DM2	1.3.5	Metadata interface	Complete interface to Grenoble metadata catalog with DC1 attributes	4/1/02	
GM-DM3	1.3.3	Job submission interfaces	Functional extension of Magda with Grappa interface	9/1/02	
GM-DM4	1.3.3	Data set regeneration	Data set regeneration using Virtual data tools developed within PPDG and GriPhyN	9/1/03	

4 Grid Enabled Data Access from Athena – Adagio

An important component of the U.S. ATLAS Grid effort is the definition and development of the layer that connects ATLAS core software to Grid middleware. Athena is the common execution framework for ATLAS simulation, reconstruction, and analysis. Athena components handle physics event selection on input, and support event collection creation, data clustering, and event streaming by physics channel on output. The means by which data generated by Athena jobs enter Grid consciousness, the way such data are registered and represented in replica and metadata catalogs, the means by which Athena event selectors query metadata, identify logical files, and trigger their delivery--all of these are the concern of this connective layer of software.

Work to provide Grid-enabled data access from within the ATLAS Athena framework is underway under PPDG auspices. Prototype implementations supporting event collection registration and Grid-enabled Athena event selectors were described at the September 2001 conference on Computing in High Energy and Nuclear Physics in Beijing (cf. Malon, May, Resconi, Shank, Vaniachine, Youssef, "Grid-enabled data access in the ATLAS Athena framework," Proceedings of Computing in High Energy and Nuclear Physics 2001, Beijing, China, September 2001). An important aspect of this work is that the Athena interfaces are supported by implementations both on the U.S. ATLAS Grid testbed (using the Globus replica catalog directly), and on the European Data Grid testbed (using GDMP, a joint EDG/PPDG product).

4.1 Adagio Schedule

Table 6 Adagio work items and milestones related to GriPhyN

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-D1		Athena Grid Registration	Grid registration of Athena products	Y2-Q2	Y2-Q4
GG-D2		Athena Grid Input	Grid-aware Athena input specification	Y2-Q2	Y3-Q2
GG-D3		Athena Runtime Grid Access	Run-time access to grid-managed data	Y2-Q3	Y3-Q3
Milestones					
GM-D1		Athena Grid Registration	Registration of Athena data products in grid replica management services	6/1/02	
GM-D2		Athena Metadata Registration	Registration of Athena data products in metadata services	9/1/02	
GM-D3		Athena Logical-file-based input	Logical-file-based input specification in Athena Job Options	9/1/02	
GM-D4		Athena Grid Event Selection	Grid-enabled Athena event selection services	3/1/03	

5 Grid User Interface – Grappa

Grappa is an acronym for Grid Access Portal for Physics Applications. This work supports U.S. ATLAS Grid WBS 1.3.9 (Distributed Analysis Development) work breakdown deliverables. The preliminary goal of this project was to provide a simple point of access to Grid resources on the U.S. ATLAS Testbed. The project began in May 2001.

5.1 Grid Portals

While there are a number of tools and services being developed for the Grid to help applications achieve greater performance and functionality, it still takes a great deal of effort and expertise to apply these tools and services to applications and execute them in an everyday setting. Furthermore, these tools and services rapidly change as they become more intelligent and more sophisticated. All of this can be especially daunting to Grid application users who are mostly interested in performance and results but not necessarily the details of how it is accomplished. One approach that has been used to reduce the complexity of executing applications over the Grid is a *Grid Portal*, a web portal by which an application can be launched and managed over the Grid⁶. The goal of a Grid Portal is to provide an intuitive and easy-to-use web (or optionally an editable script) interface for users to run applications over the Grid with little awareness about the underlying Grid protocols or services used to support their execution⁷.



5.2 Grappa Requirements

5.2.1 Use Cases

In order to understand submission methods and usage patterns of ATLAS software users, information (specifications of environment variables, operating system, memory, disk usage, average run time, control scripts, etc.) will be collected from physicists and used to formulate scenario documents, understandable by physicist and non-physicist alike. Those scenario documents will guide the further design of our Grid Portal for the submission and management of ATLAS physics jobs. One such scenario has been developed for ATLSIM⁸, the Geant3-Fortran based full simulation of the ATLAS detector. Others will be developed to provide a complete understanding of how ATLAS users will want to use the Grid.

Our initial analysis has shown that in addition to job launch, the portal must provide the ability to enter and store parameters and user annotations (notes, images, graphs) for re-use, single point authentication, real-time viewing of output and errors, and the ability to interface with mass storage devices and new Grid tools as they become available. Grappa will satisfy most of the requirements by integrating existing technologies and making them accessible via a single user interface. Tools such as the Network Weather Service, Prophesy, NetLogger are examples of existing software that Grappa will use for job management, as well as GriPhyN tools for coherent data management (Grid WBS 1.3.3.5), data distribution (Grid WBS 1.3.3.7), and data access management (Grid WBS 1.3.3.9).

Conceptually, Grappa lets physicists easily submit requests to run high throughput computing jobs on either on simple Grid resources (such as remote machine) or more advanced Grid resources such as a Condor scheduling system. Job submission allows simple parameter entry and automatic variation. Users interact from scripts or Web interfaces, and can specify resources by name or requirements. Application monitoring and job output logs are returned to the script or Web browser. A major goal of Grappa is to allow users to manage ATLAS jobs and data with an interface that does not change, while the underlying Grid tools and resources are developed within GriPhyN.

5.3 Grappa and Existing Tools

5.3.1 XCAT Science Portal

One Grid Portal effort underway at the Extreme! Computing Laboratory at Indiana University is the XCAT Science Portal which provides a script-based approach for building Grid Portals. An initial prototype of this tool has been developed to allow users to build personal Grid Portals and has been demonstrated with several applications. A simplified view of the architecture is illustrated in the Figure below. Following is a brief description.

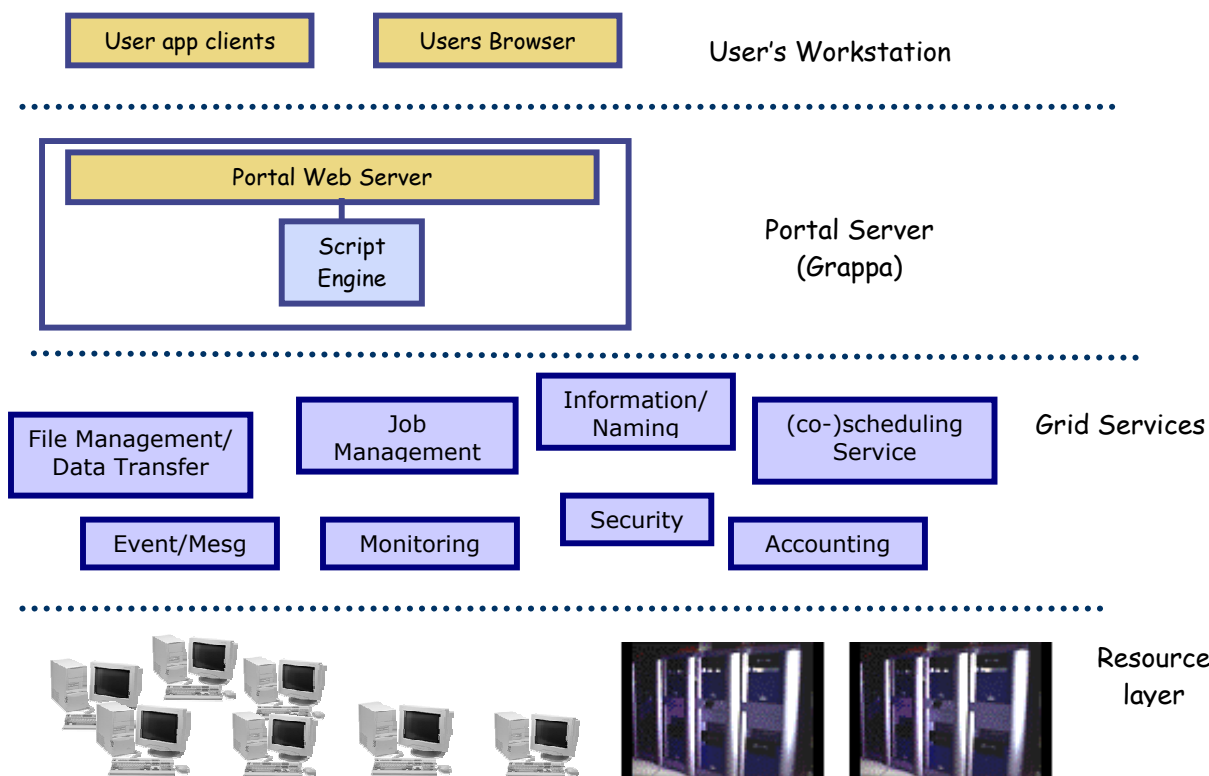


Figure 5-1 XCAT Science Grid Portal Architecture

Currently, a user authenticates to the portal using a GSI credential; a proxy credential is then stored so that the portal can perform actions on behalf of the user (such as authenticating jobs to a remote compute resource). The user can access any number of *active notebooks* within their notebook database. An active notebook encapsulates a session and consists of HTML pages describing the application, forms specifying the job's configuration, and Java Python scripts for controlling and managing the execution of the application. These scripts interface to Globus services in the GriPhyN Virtual Data Toolkit and have interfaces following the Common Component Architecture (CCA) Forum's specifications, which allows them to interact with and be used in high-performance computation and communications frameworks⁹. For a fuller description of the diagram and the XCAT Science Portal, see Ref. [6].

Using the XCAT Science Portal tools, Grappa is currently able to use Globus credentials to perform remote execution, store user's parameters for re-use or later modification, and run ATLSIM and Athena – ATLFAS^T based on the current scenario documents.

5.4 XCAT Design Changes for Grappa

Grappa will continue to build on top of the XCAT Science Portal technology, with interfaces added to the tools and services being developed by GriPhyN and other data Grid projects. While the requirements of ATLAS applications continues to be assessed (see Section 5.1), the


underlying XCAT Science Portal will be redesigned to follow a cleaner three-tier architecture, partitioned as indicated by dotted lines in Figure 5.1.

Grid Services are to be separated from the *Grid Portal* giving greater flexibility to integrate new tools as they become available, tools such as Magda, described in Section 3, and monitoring systems described in Section 6. Based on initial considerations of design requirements, the following other types of Grid Services requirements are likely candidates.

- **Job Configuration Management:** Service for storing parameters used to execute a job and user annotations for re-use (Grid WBS 1.3.5.1). This feature is currently implemented in the current XCAT Science Portal but will need to be redesigned as a Grid Service in order to facilitate sharing of job configurations among users.
- **Authentication Service:** The ability to authenticate using Grid credentials. The XCAT Science Portal currently supports a GSI and MyProxy interface for this.
- **File Management:** Service to stage input files (Grid WBS 1.3.3.9), interface with mass storage devices (Grid WBS 1.3.3.12), access replica catalog tools, etc. This will likely be the combination of several Grid Services. For example, Magda provides replica catalog access and GridFTP can be used to stage input files.
- **Monitoring:** Stores status messages, output, and errors in real time such that they can be retrieved and/or pushed to Grappa and then displayed to the user. The XCAT Science Portal can currently interface to the XEvent service (also developed by the Extreme! Computing Lab). Other monitoring services as those described in Sections 6 are likely to be accessed as well.

A second XCAT Science Portal redesign goal is support of multi-user access to the Grid Portal so that each user does not have to maintain their own web portal server but can still manage their own data separately from other users. A third goal more sophisticated parameter management interfaces, since ATLAS applications are controlled by a large number of parameters.

5.5 Grappa and Virtual Data

Grappa could be well placed to be a user interface to virtual data and in this role is similar to the NOVA work begun at Brookhaven Lab on “algorithm virtual data”, AVD¹⁰ and which will be pursued in Magda development. If virtual data with respect to materialization is to be realized, a data signature fully specifying the environment, conditions, algorithm components, inputs etc. required to produce the data must exist. These will be cataloged somehow (Virtual Data Language), somewhere (Virtual Data Catalog), or the components that make them up are cataloged and a data signature is a unique collection of these components constituting the 'transformation' needed to turn inputs into output. Grappa could then interface to the data signature and catalogs and allow you to 'open' a data signature and view it in a comprehensible , edit it, run it, etc. Take away the specific input/output data set(s) associated with a particular data signature and you have a more general 'prescription' or 'recipe' for processing inputs of a given type under very well defined conditions, and it will be very interesting to have

catalogs of these -- both of the 'I want to run the same way Bill did last week' variety and 'official' or 'standard' prescriptions the user can select from a library.

5.6 Grappa Schedule

Table 7 Grappa work items and milestones

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-I1		Multi-user Portal	Extend portal server to multi-user	Y2-Q1	Y2-Q4
GG-I2		Use scenarios	Analyze user scenario documents	Y2-Q1	Y2-Q1
GG-I3		Job launch extension	Grappa using other launch tools	Y2-Q2	Y2-Q4
GG-I4		Evaluation of Grid based file management systems (GFMS)	Continued evaluation and monitoring of developments in Grid-based file management systems such as Magda, SRB, Globus replica catalog service	Y2-Q2	Y2-Q4
GG-I5		Implement interface to GFMS	Design, prototype, implement and test Grappa interface to suitable GFMS	Y2-Q2	Y2-Q4
GG-I6		Condor DAGman interface	Implement DAGman functionality	Y2-Q2	Y2-Q4
GG-I7		Parameter management	Explore large parameter set management	Y2-Q1	Y2-Q4
Milestones					
GM-I1		Condor-G functionality	Demonstrate use of Condor-G from Grappa	7/1/02	
GM-I2		GFMS Evaluation	First evaluation of GFMS complete	4/1/02	
GM-I3		GFMS Interface	First release of GFMS interface with Grappa	7/1/02	

6 Performance Monitoring and Analysis

Performance monitoring and analysis is an important component necessary to insure efficient execution of ATLAS applications on the Grid. This component entails the following:

- ❑ Instrumenting ATLAS applications to get performance information such as event throughput and identifying where time is being spent in the application
- ❑ Installing monitors to capture performance information about the various resources (e.g., processors, networks, storage devices)
- ❑ Developing higher level services to take advantage of this sensor data, for example, to make better resource management decisions or to be able to visualize the current testbed behavior

- Developing models that can be used to predict the behavior of some devices or applications to aid in making decisions when more than one option is available for achieving a given goal (e.g., replication management)

Many tools will be used to achieve the aforementioned goals. Further, the performance data will be given in different formats, such as log files or data stored in databases. Additional tools will be developed to analyze the data in the different formats and visualize it as needed. The focus of this work will be on the U.S. ATLAS testbed.

We are leading the joint PPDG/GriPhyN effort in monitoring to define the use cases and requirements for a cross-experiment testbed. As part of the joint effort we have been gathering use cases to define requirements for the information system needed for a Grid-level information system, in part to answer questions such as these. The next step of this work will be to define a set of sensors for every facility to install, and to develop and deploy the sensors and their interface to the Globus Metacomputing Directory Service (MDS) and other components as part of the testbed. The services, needed to make execution on compute Grids transparent, will also be monitored. Such services include those needed for file transfer, access to metadata catalogs, and process migration.

Details about the Grid-level monitoring are given in Section 6.1 along with information about the visualization of this data using GridView. In addition, we have been evaluating and installing sensors to capture the needed data for our testbed facilities internally, and determining what information should be shared at the Grid level, and the best ways to do this, as detailed in Section 6.2. At the application level, much work has been done with Athena Auditor services to evaluate application performance on the fly, as described in 6.3. Section 6.4 discusses some higher level services work in prediction. Section 6.5 discusses work plans for a grid telemetry system.

6.1 Grid-level Monitoring

At the Grid-level, several different types of questions are asked of an information service. This can include scheduling-based questions, such as what is the load on a machine or network or what is the queue on a large farm of machines, as well as data-access questions like – where is the fastest repository I can download my file from?

We are defining a standard set of sensors to be installed on the testbed in order to address these types of questions, and to interface with the Globus information service, MDS. In addition, we are developing additional sensors as needed to conglomerate data on local farms, for example, and advertise this summary data to the grid.

One area that has received a great deal of attention in the group already is monitoring network resources. There are two main types of network sensors – passive sensors that sniff on a network connection or active sensors that create network traffic to obtain information about network bandwidth, package loss, and round-trip time. There are many tools available for network monitoring, iperf, Network Weather Service, pingER and so on. We need to support the deployment of these testing and monitoring tools and applications, in association with the HENP network working group initiative, so that most of ATLAS major network paths can be

adequately monitored. The network statistics should be included in Grid information service so that Grid software can choose the optimized path for accessing the virtual data.

In order to make better use of the data advertised by various sensors or tools, GridView was developed at the University of Texas at Arlington (UTA) to monitor the U.S. ATLAS Grid, first released in March, 2001. GridView provides a snapshot of dynamic parameters like CPU load, up time, and idle time for all Testbed sites. The primary web page, a snapshot shown in Figure 6-1 below, can be viewed at:

<http://heppc1.uta.edu/kaushik/computing/Grid-status/index.html>

GridView has gone through two subsequent releases. First, in summer 2001, MDS information from GRIS/GIIS servers was added. Not all Testbed nodes run a MDS server. Therefore, the front page continues to be filled using basic Globus tools. MDS information is provided in additional pages linked from this front page, where available.

Recently, a new version of GridView was released after the beta release of Globus 2.0 in November 2001. The U.S. ATLAS Testbed incorporates a few test servers running Globus 2.0 as well as every Testbed site running the stable 1.1.x version. GridView provides information about both types of systems integrated in a single page. Globus has changed the schema for much of the MDS information with the new release, but GridView can query and display either type. In addition, a MySQL server is used to store archived monitoring information. This historical information is also available through GridView.

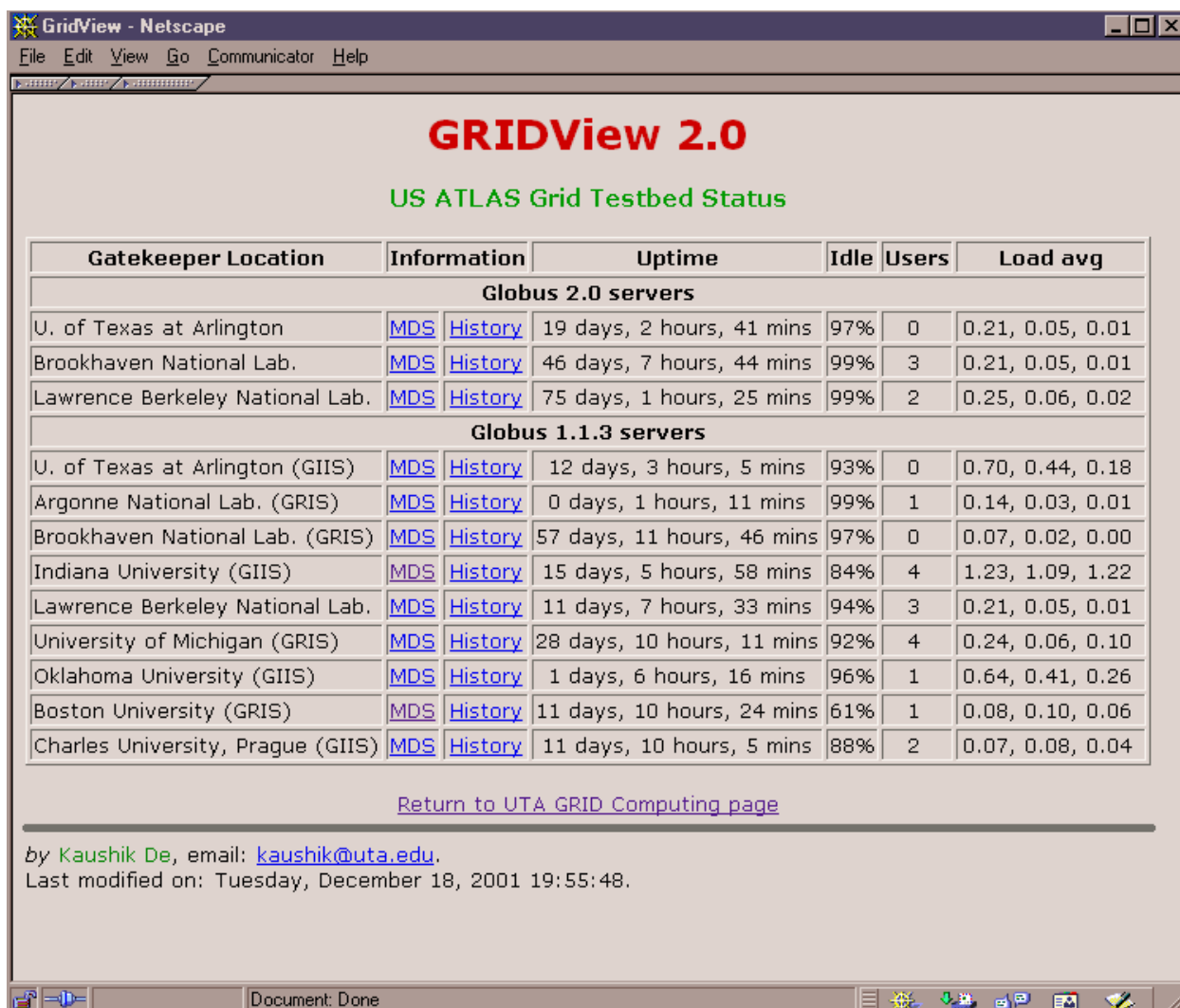


Figure 6-1 GridView display of U.S. ATLAS Grid Testbed

6.2 Local Resource Monitoring

There are different monitoring needs on a wide-area (grid-level) system than on a local system. Primarily in this is the need for local data to be summarized up to the grid level system for scalability purposes. Dantong Yu at Brookhaven has been leading the effort in local resource monitoring for the U.S. ATLAS testbed.

The different resources used to execute ATLAS applications will be monitored to aid in accessing different options for the virtual data. Initially, the following resources will be monitored with different tools: system configuration, network, host information and important processes:

- **System Configuration:** Monitoring systems should perform a software and hardware configuration survey periodically and obtain the information on what software (version, producer) are installed on this system, what hardware is available. This data is then collected together for an entire set of local resources and advertised to the grid level to be used, for example, by a Grid scheduler to choose the right system environment for the system-dependent ATLAS applications.
- **Host/Device Monitoring:** host information includes CPU load, Memory load, available memory, available disk space, and average disk I/O time. Once summarized for an entire farm of machines or advertised out as is for single resource, this information will help Grid scheduler and Grid user to choose computing resource to run ATLAS applications intelligently. In addition, an ATLAS facility manager can use this information for site management. The necessary information for Grid computing will be identified and deployed at ATLAS testbed.
- **Process Monitoring:** Process sensors monitor the running status of a process, such as number of this type of processes, number of users, queue lengths, etc.. one use of these sensors to have a threshold set up to trigger an alarm when the threshold is reached to prevent overloading system resources or help recover the system from failure.

The local resource monitoring effort is currently being coordinated with PPDG, GriPhyN, iVDGL, EU DataGrid and other HENP experiments to ensure that the local resource monitoring infrastructures satisfy the needs of Grid users and Grid applications.

6.3 Application Monitoring

The ATLAS applications will be instrumented at various levels to obtain performance information on how much time is spent with between accesses to data and used with different data.

First, some of the Athena libraries will be instrumented so to get detailed performance information about file access and file usage. For the case when the instrumentation overhead is small, the libraries can be automatically used when specified in a user's job script. For the case when the instrumentation overhead is large, the instrumented libraries must be specified by the user; such libraries will not be used by default.

Second, the Athena auditors will be used to obtain performance information. The auditors provide high-level information about execution of different Athena algorithms. Auditors are executed before and after the call to each algorithm, thereby providing performance information at the level of algorithm execution. Currently, Athena includes auditors to monitor the CPU usage, memory usage and number of events for each Athena algorithm. Athena also includes a Chrono & Stat service to profile the code (Chrono) and perform statistical monitoring (Stat). Hence, Athena will be instrumented at both the algorithm and libraries levels to obtain detailed performance data.

6.4 Performance Models

The trace data found in log files and performance databases (such as MDS and Prophecy) will be used to develop analytical performance models that can be used to evaluate different options related to access to virtual data. In particular, various techniques will be used such as curve fitting and detailed analysis and modeling of the core ATLAS algorithms. The models will be refined as more performance data is obtained. The models can be used to evaluate options such as is it better to obtain data from a local site for which it is necessary to perform some transformations to get the data in the desire format or access the data from remote sites for which one needs to consider the performance of given resources such as networks and the remote storage devices. The analytical models would be used to evaluate the time needed for the transformation based upon the system used for execution.

6.5 Grid Telemetry

A distinction is made between grid *instrumentation* and grid *telemetry*. At the fabric level, instrumented devices such as network components (data switches and routers) produce data for status and monitoring purposes. For example, data flowing from these devices is captured and used in problem management situations by Network Operation Centers (NOC) and for on-line and archival monitoring. Such data be of a temporal nature and may signal critical events such as equipment failure, bottlenecks and congestion, or the data may report performance measures such as bandwidth utilization along a given network link.

Extending the concept, telemetry data can be captured and sent to/from various sources for monitoring and input to resource allocation algorithms. At the application level, “counters” which record, for example, numbers of events in a production system for a particle physics simulation can be collected from distributed sources to be used for high level tracking and monitoring. At the middleware level, workflow managers and distributed batch systems such as Condor¹¹ may require (or provide) telemetry data to improve efficiency of operation or to take advantage of new resources as they become available. At lower levels, services indicating CPU utilization, the status of authentication services, host monitoring, data transfer (I/O load indicators), cache and archive storage utilization need to be collected to provide an information basis for job planning and resource estimation.

Several groups have developed toolkits which either produce or provide instrumentation hooks for grid telemetry data. The Internet End-to-end Performance Monitoring (IEPM) Group¹² has developed a set of tools to monitor data collection, site connectivity, and tools for monitoring packet loss and response time of registered sites within the network. The Indiana University Network Administration Suite is a collection of programs developed for the maintenance and management of IU campus networks as well as the Abilene, TransPAC, and STAR TAP networks¹³. The Netlogger¹⁴ toolkit developed at NERSC provides a message passing library that enables real-time diagnosis of problems in complex high-performance distributed systems. The tool has been successfully used to debug low throughput or high latency problems in distributed applications. The system includes tools for generating precision event logs that can be used to provide detailed end-to-end application and system level monitoring, and tools for visualizing log data to view the state of the distributed system in real time. Open source Linux

cluster management toolkits such as NPACI Rocks¹⁵ provide monitoring data which can be useful if sourced to remote monitoring systems. What is missing from these toolkits is the surrounding infrastructure to collect, archive, and manage the information in formats suitable for high level problem management, diagnosis, and resource information systems.

A separate ITR proposal was funded to develop a grid telemetry data acquisition system which intends to build on these advances by providing an integrated collection system for monitoring, problem diagnosis and management, and resource decision making algorithms operating in a grid environment. Since telemetry data acquisition systems will be linked closely with their sources, it is important that development of such a system be made in close communication with application, middleware and fabric developers and engineers. Development of such a system within the context of particle physics data grids and research projects such as petabyte-scale virtual data grid research by GriPhyN provides this opportunity. This work, which will be coordinated closely with the monitoring activities of GriPhyN, PPDG, and others participating in the joint monitoring group, will thus support GriPhyN and iVDGL laboratory operations.

6.5.1 Prototype Grid Telemetry System

A prototype grid telemetry acquisition system is shown schematically in the Figure 6-2. Telemetry data is collected and organized into “pools”, which could be distributed to provide redundancy and scalability. The pools consist of database servers and storage area networks connected to the external network over fast links, and some may have access to archival (tape) storage systems. The data residing in these pools would be accessible with a variety of tools, including web based applications for visualization and API's written in Java, Perl or Python. Netlogger may be used as a message passing service for the system.

Each layer in the distributed grid may be instrumented to source telemetry data. Instrumented applications in particle physics may report event statistics, error conditions, and performance data. Data recorded by the server can be queried by the planning, estimation, and execution layer to optimize throughout performance. Archival, transport and data caches can report status and other performance data to the servers, again to be used by the upper two layers. In addition, security services can be queried, and policy decisions for specific applications or grid users using information logged by the server. The fabric can be continuously monitored and both real-time and historical data for host status/performance, network performance, data cache capacities, for example, can be archived by the system.

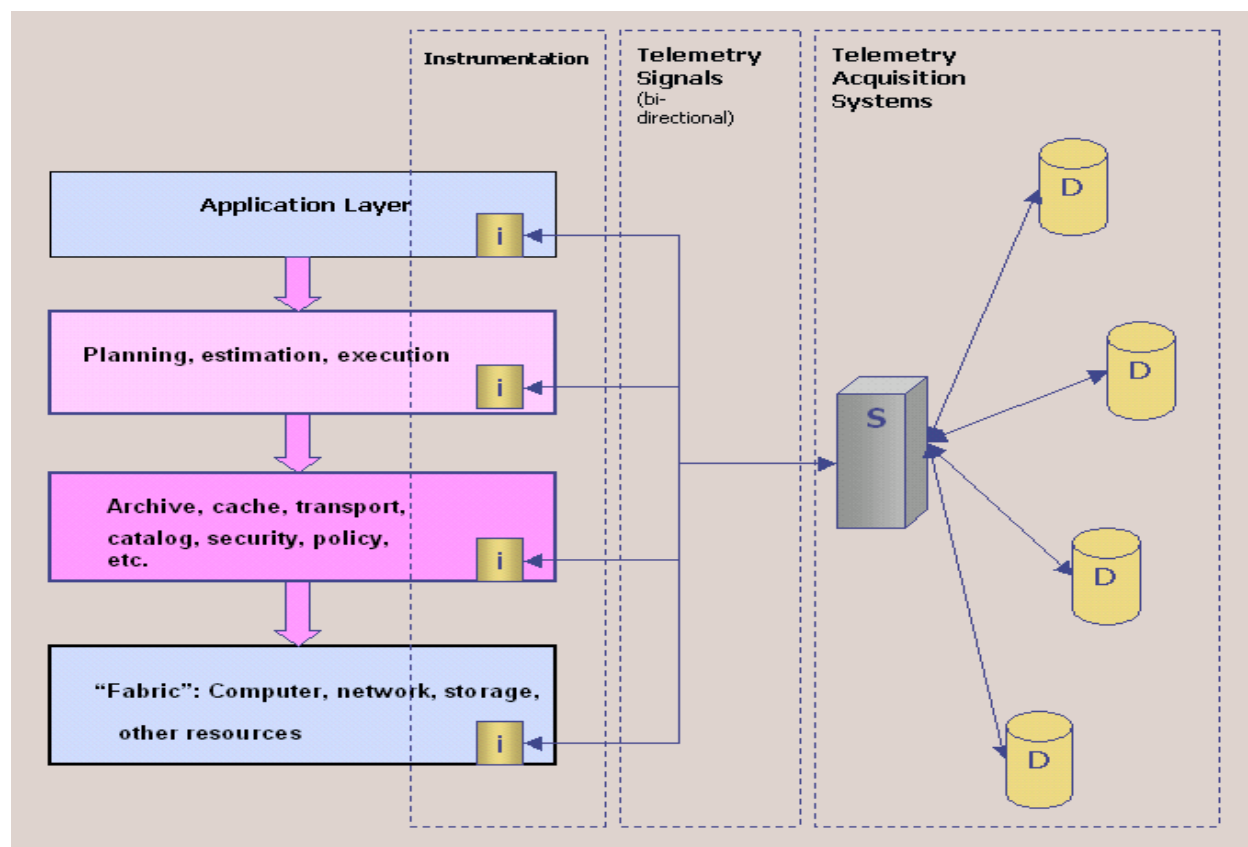


Figure 6-2 Components of a grid telemetry acquisition system. Instrumented modules within the grid application, middleware, and fabric levels (denoted by modules “i”) transmit and receive telemetry signals from a server “S” providing access databases “D”.

6.5.2 Telemetry Program of Work

The main outline of work is the following:

1. Collect monitoring and resource decision making requirements from core application physicists and grid middleware developers.
2. Identify and evaluate existing toolkits which source grid telemetry data.
3. Design the high level architecture for the grid telemetry data acquisition system and create technical design specification.
4. Prototype the design.
5. Implement grid telemetry data acquisition system. Facilities at Indiana University will be used. The hardware requirements are for dedicated database servers and storage area networks to provide high performance access to telemetry databases.
6. Instrument application level monitors for ATLAS core software.
7. Provide a resource service for grid and application developers.

The work will be carried out within the context of the U.S. ATLAS, GriPhyN, iVDGL, and international CERN testbeds.

6.6 Monitoring Schedule

Table 8 Monitoring work items and milestones

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-M1		Evaluation	Initial evaluation of Grid monitoring services and requirements	Y2-Q1	Y2-Q4
GG-M1.1		Requirements analysis and specification	Requirements will be built from use case scenarios		
GG-M1.2		Evaluation of existing monitoring tools	For each type of local monitoring information, we will evaluate 2~3 monitoring tools.	Y2-Q1	Y2-Q1
GG-M1.3		Identify and select necessary mentoring tools.	Some tools may be modified and developed as part of this effort if they are not addressed by other work groups and not available commercially.	Y2-Q2	Y3-Q3
GG-M2		Tools	Development of tools, integration of identified monitoring services into Grid information service, etc.	Y3-Q1	Y4-Q4
GG-M2.1		Tool deployment Phase I	Deploy the required tools for testing at single sites	Y2-Q1	Y2-Q1
GG-M2.2		Tool Deployment – Phase II	Refine tool suite as needed, deploy on two sites with some feedback used to make some decisions; incorporate tools with information databases	Y2-Q1	Y2-Q2
GG-M2.3		Tool Deployment – Phase III	Incorporate tools into for inter-site monitoring	Y2-Q2	Y2-Q3
GG-M2.4		Tool Deployment – Phase IV	Incorporate tools into for inter-site monitoring	Y2-Q3	Y2-Q4
GG-M3		GridView	Grid information views	Y2-Q1	Y2-Q4
GG-M3.1		GridView – Phase I	Setup hierarchical GIS server based on Globus 2.0	Y2-Q1	Y2-Q2
GG-M3.2		GridView – Phase II	Develop graphical tools for better organization of monitored information.	Y2-Q3	Y2-Q4
GG-M4		Grid Telemetry		Y2-Q2	Y3-Q4
GG-M4.1		Requirements gathering	Collect monitoring and resource decision making requirements from core application physicists and grid middleware developers.	Y2-Q2	Y2-Q3
GG-M4.2		Evaluation	Identify and evaluate existing toolkits which source grid telemetry data.	Y2-Q2	Y2-Q3
GG-M4.3		Design	Design the high level architecture for the grid telemetry data acquisition system and create	Y2-Q4	Y2-Q4

			technical design specification		
GG-M4.4		Prototype	Prototype the design.	Y2-Q4	Y3-Q1
GG-M4.5		Implement	Implement grid telemetry data acquisition system. Facilities at Indiana University will be used. The hardware requirements are for dedicated database servers and storage area networks to provide high performance access to telemetry databases.	Y3-Q1	Y3-Q2
GG-M4.6		Instrument	Instrument application level monitors for ATLAS core software	Y2-Q2	Y2-Q3
GG-M4.7		Production service	Provide a resource service for grid and application developers.	Y2-Q2	Y2-Q3
Milestones					
GM-M1		Monitoring Tool X evaluation	Evaluation of tool X completed	End Y2-Q4	
GM-M2		Deploy monitoring tools	For each type of local monitoring (network, host, configuration, important service), at least one tool should be identified and deployed at each individual ATLAS testbeds.	End Y2-Q4	
GM-M3		Construct Performance databases	Importance performance trace data should be archived in databases. Integrate the database into Grid information services.	End Y2-Q4	
GM-M4		First integration into Athena Auditor package	First test integration of GriPhyN monitor tools with Athena Auditor services	End Y2-Q3	

7 Grid Package Management – Pacman

If ATLAS software is to be smoothly and transparently used across a shifting Grid environment, we must also gain the ability to reliably define, create and maintain standard software environments that can be easily moved from machine to machine. Such environments must not only include standard ATLAS software via CMT and CVS, must also include a large and growing number of “external” software packages as well as Grid software coming from GriPhyN itself. It is critical to have a systematic and automated solution to this problem. Otherwise, it will be very difficult to know with confidence that two working environments on the Grid are really equivalent. Experience has shown that the installation and maintenance of such environments is not only labor intensive and full of potential for errors and inconsistencies, but also requires substantial expertise to install and configure correctly.

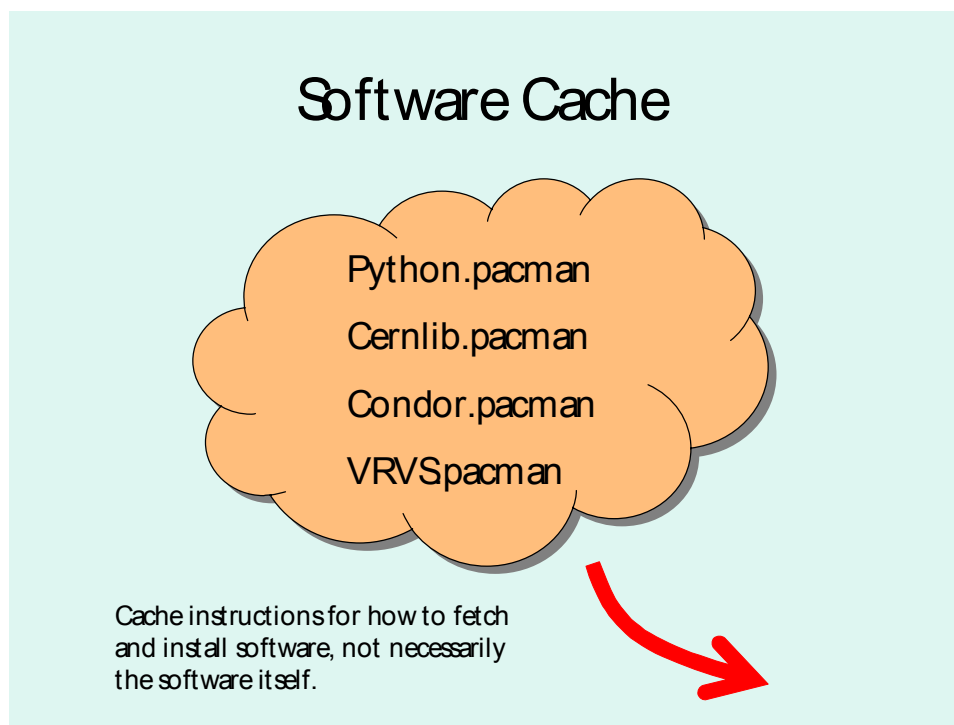


Figure 7-1 Pacman -- Package Management System components

To solve this problem we propose to effectively raise the problem from the individual machine or cluster level to the Grid level. Rather than having individual ATLAS sites work through the various installation and update procedures, we can have individual experts define how software is fetched, configured and updated and publish these instructions via “trusted caches.” By including dependencies, we can define complete named environments which can be automatically fetched and installed with one command and which will result in a unified installation with common setup script, pointers to local and remote documentation and various such conveniences. Since a single site can use any number of caches together, we can distribute the expertise and responsibility for defining and maintaining these installation procedures across the collaboration. This also implies a shift in the part of Unix culture where individual sites are expected to work through any installation problems that come up in installing third party software. The responsibility for an installation working must, we feel, be shifted to the “cache manager” who defined the installation procedure to begin with. In this way, problems can be fixed once by an expert and exported to the whole collaboration automatically.

Over the next year or so, and particularly in order to prepare for Data Challenge 1, we will use an implementation of the above ideas called “Pacman” to define standard ATLAS environments which can be installed via caches. This will include run-time ATLAS environments, full development environments and project specific user defined environments. In parallel, we will work with the VDT distribution team and with Globus to develop a second-generation solution to this problem that can be more easily integrated with the rest of the GriPhyN Grid tools.

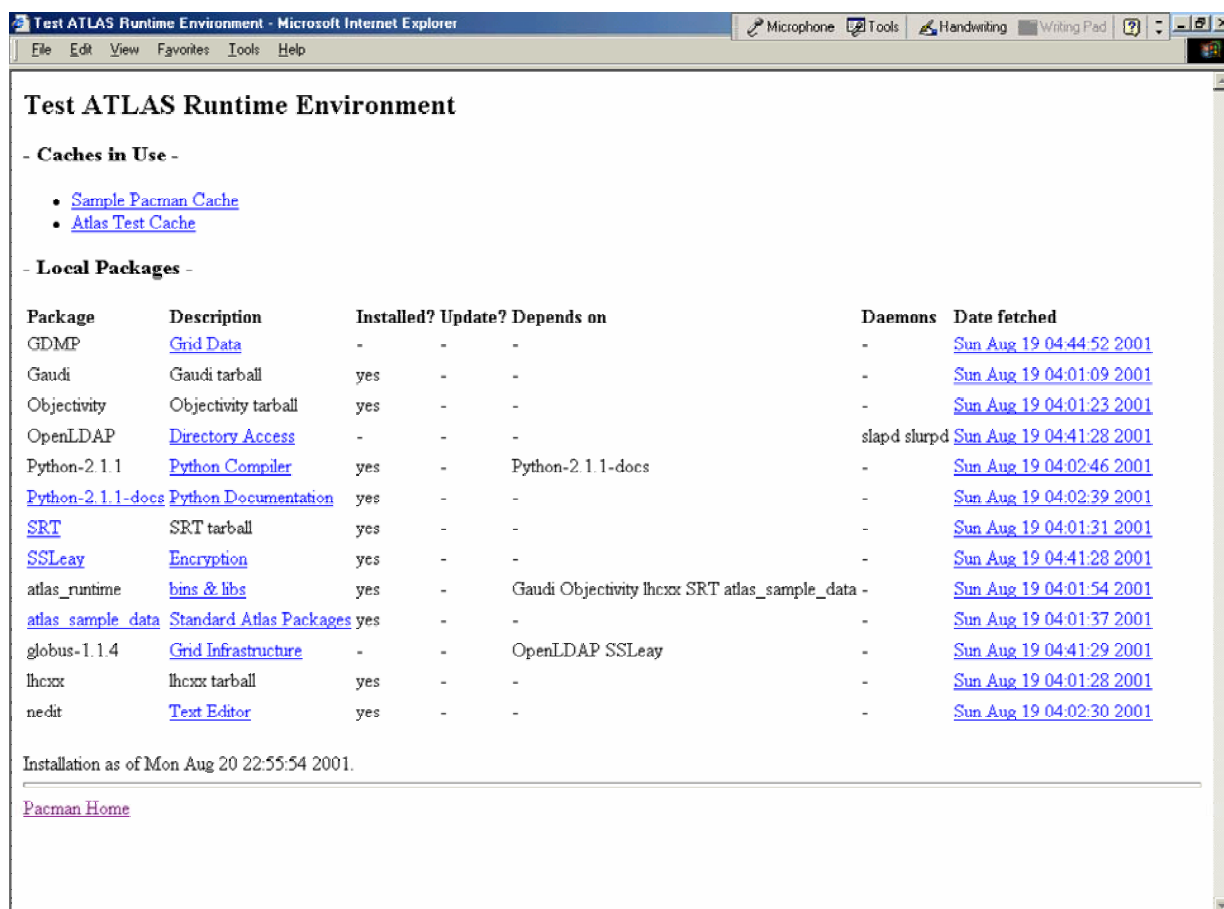


Figure 7-2 Package cache display in Pacman

7.1 Pacman Schedule

Table 9 Pacman work items and milestones

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-P1		Pacman distribution of Globus 2	Configured as needed for DC1	Y2-Q1	Y2-Q2
GG-P2		Pacman distribution of VDT 1.0	Working with Miron Livny's team	Y2-Q1	Y2-Q3
GG-P3		Feedback Pacman experience to GriPhyN CS teams	Work with GriPhyN CS teams to develop second generation solutions to grid package management	Y2-Q1	Y3-Q4
GG-P4		All 3d party software needed by Atlas distributed with Pacman		Y2-Q1	Y2-Q2
GG-P5		Pacman integrated with CMT		Y2-Q2	Y2-Q3

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-P6		Pacman more general dependences implemented		Y2-Q2	Y2-Q2
GG-P7		Caches setup at BNL, BU, Indiana, UT Arlington, LBNL, Michigan		Y2-Q1	Y2-Q1
Milestones					
GM-P1		ATLAS AFS, runtime and stand along development environments delivered with Pacman.	Single operation of full installation of ATLAS environments on Linux and Sun Solaris.	Y2-Q2	

8 Security and Accounting Issues

We will work with the existing GSI security infrastructure to help the Testbed groups deploy a secure framework for distributed computations. The GSI infrastructure is based on the Public Key Infrastructure (PKI) and uses public/private key pairs to establish and validate the identity of Grid users and services. The system uses X.509 certificates signed by a trusted Certificate Authority (CA). Presently U.S. ATLAS Testbed sites use the Argonne/Globus CA, but will begin accepting ESNet CA certificates. By using the GSI security infrastructure we will be compatible with other Globus-based projects, as well as adhering to a de-facto standard in Grid computing. We will work in close collaboration with ESNet and PPDG groups working on CA issues to establish and maintain Grid certificates throughout the testbeds. We will support and help develop a Registration Authority for ATLAS – GriPhyN users.

A related issue is the development of an authorization service for resources on the testbed. Within Globus, there is much research on-going effort¹⁶ which we will closely follow and support when these services become available.

9 Site Management Software

The LHC computing model implies a tree of computing centers where “Tier X” indicates depth X in the tree. For example, Tier 0 is CERN, Tier 1 is Brookhaven National Laboratory, and Boston University and Indiana University are “Tier 2” centers, etc. University groups are at the Tier 3 level and Tier 4 is meant to be individual machines. While the top of this tree is fairly stable, we must be able to add Tier 3 and Tier 4 nodes coherently with respect to common software environment, job scheduling, virtual data, security, monitoring and web pages while guaranteeing that there is no disruption of the rest of the tree as nodes are added and removed. To solve this problem we propose to define what a Tier X node consists of in terms of installed ATLAS and Grid software and to define how the Grid tools are connected to the existing tree. Once this is done, we propose to construct a nearly automatic procedure (in the spirit of Pacman or successors) for adding and removing nodes from the tree. Over the next year, we will gain

enough experience with the top nodes of tree of Tiers to understand how this must be done in detail. In 2002, we propose to construct the software that nearly automatically adds Tiers to the tree.

10 Testbed Development

10.1 U.S. ATLAS Testbed

The U.S. ATLAS Grid Testbed is a collaboration of ATLAS U.S. institutions that have agreed to provide hardware, software, installation support and management of collection of Linux based servers interconnected by the various U.S. production networks. The motivation was to provide a realistic model of a Grid distributed system suitable for evaluation, design, development and testing of both Grid software and ATLAS applications to run in a Grid distributed environment. The participants include designers and developers from the ATLAS core computing groups and collaborators on the PPDG and GriPhyN projects. The original (and current) members are the U.S. ATLAS Tier 1 computing facility at Brookhaven Laboratory, Boston University and Indiana University (the two prototype Tier 2 centers), Argonne National Laboratory HEP division, LBNL (PDSF at NERSC), the University of Michigan, Oklahoma University and the University of Texas at Arlington. Each site agreed to provide at least one Linux server based on Intel X86 running Red Hat version 6.x OS and Globus 1.1.x gatekeeper software. Each site agreed to host user accounts and access based on the Globus GSI x509 certificate mechanisms. Each site agreed to provide a native or AFS based access to the ATLAS offline computing environment, sufficient CPU and Disk resources to test Grid developmental software with ATLAS codes. Each site volunteers technical resource people to install and maintain a considerable variety of infrastructure for the Grid environment and developed software by the participants. In addition, some of the sites choose to make the Grid gatekeepers as gateway to substantial local computing resources via Globus job manager access to LSF batch queues or Condor pools. This has been facilitated and managed by bi-weekly teleconference meetings over the past 18 months. The project began with a workshop¹⁷ on developing a GriPhyN – ATLAS testbed at Indiana University in May 2000. A second testbed workshop¹⁸ was held at the University of Michigan in February 2001.

The work of the first year included installation and operation of an eight node Globus 1.1.x Grid; installation and testing of components of the U.S. ATLAS distributed computing environment, development and testing of PDSF developed tools. These included Magda, GDMP, and alpha versions of the Globus DataGrid Tool sets. Testing and evaluation of the GRIPE account manager¹⁹, the development and testing of network performance measurement and monitoring tools. The development, installation and routine use of Grid resource tools e.g. GridView. The development and testing of new tool for distribution, configuration and installation of software: Pacman. The testing of the ATLAS Athena code ATLFast writing and reading to Objectivity databases on the testbed gatekeepers; testing and preparations for installation of Globus 2.0 and associated DataGrid tools to be packaged in the GriPhyN VDT1.0; preparations and coordination with the European DataGrid testbed, and coordination with the International ATLAS Grid project. The primary focus has been on developing infrastructure and tools.

The goals of the second year will include: Continuing the work on infrastructure and tools installation and testing. A coordinated move to a Globus 2.0 based Grid. Providing a reliable test environment for PPDG, GriPhyN and ATLAS core developers. The adoption and support of a focus on ATLAS application codes designed to exploit the Grid environment and this testbed in particular. A principal mechanism will be the full participation in the ATLAS Data Challenge 1 (DC1) exercise. This will require the integration of this testbed into the EU DataGrid and CERN Grid testbeds. During the second half we expect to provide a prototype Grid based production data access environment to the simulation data generated as part of DC1, thus a first instance of the U.S. based distributed computing plan for U.S. offline analysis of ATLAS data. To achieve these goals we will evolve the US testbed into two pieces: an eight site prototype-production grid (stable, user-friendly, production and services oriented) and (4-8 site) test-bed grid (with traditional test-bed properties for developmental software and quick turn-around reconfiguration etc).

10.2 iVDGL

The iVDGL project will provide the computing platform upon which to evaluate and develop distributed Grid services and analysis tools developed by GriPhyN. Two ATLAS – GriPhyN institutions will develop prototype Tier 2 centers as part of this project, Indiana University and Boston University. Resources at those facilities will not only support ATLAS specific applications, but also the iVDGL/GriPhyN collaboration at large, both physics applications and CS demonstration/evaluation challenges. In addition, other sites within the iVDGL, domestic and international, will be exploited were possible for wide area job execution using GriPhyN developed technology.

10.3 Infrastructure Development and Deployment

Below are some specific software components which need to be configured on the Testbed.

Testbed configuration during Year 2:

- ❑ VDT1.0 (Globus 2.0Beta, Condor 6.3.1, GDMP 2.0)
- ❑ ATLAS Software releases 2.0.0 and greater
- ❑ Magda
- ❑ Objectivity 6.1
- ❑ Pacman
- ❑ Test suite for checking proper installation
- ❑ Documentation, web-based, for the Testbed configuration at each site
- ❑ The above packaged with Pacman

We will begin by deploying VDT services, ATLAS software, and ATLAS required external packages on a small number of machines at 4 to 8 sites. At each site, skilled personnel are identified as points of contact. These are: ANL (May), BU (Youssef), BNL (Yu), IU (Gardner) in first 3 months (required), with UTA, NERSC, U of Michigan, and OU following as their effort allows. Additional parts of the work plan include:

1. Identify a node at CERN to be included in early testbed development. This will include resolution of CA issues, and accounts. May leads.
2. Define simple ATLAS application install procedure, neatly package up a simple example using Pacman, including documentation, with simple run instructions and a readme file. Sample data file and a working Athena jobs are needed. Ideally, several applications will be included. Shank, Youssef, and May lead.
3. Provide an easy setup for large scale batch processing. This will include easy account and certificate setup, disk space allocations, and access to other site specific resources. Ideally this will be done with a submission tool, possibly based on Grappa or included within Magda. Wenaus, Smallen lead.

10.4 Testbed Schedule

Table 10 Testbed work items and milestones

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-T1		GT1 testbed	Establish a 4-8 site testbed in parallel	Y2-Q1	Y2-Q1
GG-T2		Migrate 8 site testbed (GT1.1.x) to GT2	Establish proto-production US ATLAS Grid of 8 sites; uniform installation of VDT 1.0 and other Grid tools.	Y2-Q2	Y2-Q3
GG-T3		CA migration and global integration	Migrate both testbed and proto-production sites to ESnet CA and integration with EU DG, CERN, and other ATLAS Grids	Y2-Q2	Y2-Q3
GG-T4		DC1 participation	Integration with backend compute and data services; execute DC1 tests		
GG-T5		DC1 data services	Establish and execute production services for DC1 data analysis on the proto-production Grid		
GG-T6		SC demo preparations	Configuration and preparations for SC 2002 demonstrations	Y2-Q4	Y3-Q1
Milestones					
GM-T1		GT1 testbed	Demonstration GT1 testbed to be operational DC1 development	10/01/01	
GM-T2		VDT 1.0	VDT 1.0 deployed on all sites	1/1/02	
GM-T3		CERN Testbed node	Installation, configuration of a dedicated Grid Testbed node at CERN	1/1/02	
GM-T4		GT2 testbed	Demonstration GT2 testbed to be operational for DC1 analysis	7/1/01	
GM-T5		SC demo	SC demo preparations complete	11/1/01	

11 ATLAS – GriPhyN Outreach Activities

We plan to join GriPhyN and iVDGL outreach efforts with a number of on-going efforts in high

energy physics, including, the ATLAS Outreach committee and QuarkNet.

Within GriPhyN – iVDGL, Hampton University will be building a Tier 3 Linux cluster as part of the iVDGL Outreach effort. HU is also a member of ATLAS and is heavily involved in construction of the Transition Radiation Tracking detector for ATLAS.

Specific outreach work items:

1. Provide ATLAS liaison and support for the GriPhyN Outreach Center²⁰.
2. Provide support and consultation for installation of GriPhyN VDT and ATLAS software at HU.
3. Interact with Hampton University students and faculty running ATLAS applications on the Grid.
4. Support HU and other iVDGL institutions in establishing a GriPhyN – iVDGL QuarkNet educational activities.

11.1 Outreach Schedule

Table 11 Outreach work items and milestones

GriPhyN Code	ATLAS Grid WBS	Name	Description	Start	End
GG-O1	NA	Web support	Provide web based information for ATLAS – GriPhyN activities for education and outreach purposes	Y2-Q1	Y3-Q4
GG-O2	NA	VDT support	Provide support for VDT installation, guidance at Hampton University	Y2-Q3	Y3-Q4
GG-O3	NA	QuarkNet	Interact with EO outreach faculty developing GriPhyN QuarkNet programs for high school teachers and students	Y2-Q3	Y3-Q4
GG-O4	NA	ATLAS Software	Provide support to HU and other outreach institutions requiring assistance with ATLAS software installation and support.	Y2-Q3	Y3-Q4
Milestones					
GM-O1	NA	Web page	GriPhyN – ATLAS outreach webpage complete	Y2-Q4	
GM-O2	NA	ATLAS job submission	Demonstration of ATLAS Monte Carlo generation, reconstruction, analysis codes executing by students and faculty located at outreach institutions using Grappa interface	Y2-Q4	

12 Summary, Challenge Problems, Demonstrations

Below we give a summary and schedule of GriPhyN – ATLAS goals, challenge problems, and demonstrations.

12.1 ATLAS Year 2 (October 01 – September 02)

12.1.1 Goals Summary

As discussed previously, during Year 2 ATLAS Data Challenges 0 and 1 occur and ATLAS will build up a large volume of data based on the most current detector simulation model and processed with newly developed reconstruction and analysis codes. This is an important opportunity for GriPhyN as there will be a demand throughout the collaboration for distributed access to the resulting data collections, particularly the reconstruction and analysis products which contain file sets suitable for analysis. These will occur during the second phase of DC1, ending sometime in July 2002. In close collaboration with PPDG we will integrate VDT data transport and replication tools, with particular focus on reliable file transfer tools, into a distributed data access system serving the DC data sets to ATLAS users. We will also use on-demand regeneration of DC reconstruction and analysis products as a test case for virtual data by materialization. These exercises will test and validate the utility of Grid tools for distributed analysis in a real environment delivering valued services to end-users.

Collaboration with the International ATLAS Collaboration, and the LHC experiments overall is an important component of the subproject. In particular, developing and testing models of the ways ATLAS software integrates with Grid middleware is a critical issue. The international ATLAS collaboration, with significant U.S. involvement, is responsible for developing core software and algorithms for data simulation and reconstruction. The goal is the successfully integrate Grid middleware with the ATLAS computing environment in a way that provides a seamless Grid-based environment used by the entire collaboration.

12.1.2 Challenge Problem I: DC Data Analysis

Data Challenge 1 (February 2002 - July 2002) involves producing 1% of the full-scale solution using existing core ATLAS software. The execution will run in a traditional, linear fashion without Grid interactions. Event generation (using the PYTHIA generator package from the Lund group) will be invoked from the Athena framework, while the Geant3-based detector simulation will use the Fortran-based program. The result will be data sets that are of interest to users in general, generating 10^7 events using O(1000) PC's, with a total data size of 25-50 TB.

Planning and execution of CP I will involve:

1. Tagging the data sample with physics generator metadata tags, and storage in a metadata file system for subsequent collection browsing. The Grenoble group is leading this effort; the interface with Magda is being done by PPDG.
2. Serving the data (and metadata) using Grid infrastructure file access. At a bare minimum, a well organized website will be used to supply first time ATLAS users a portal to the DC collections. A more advanced solution, similar to present Magda functionality,

which incorporates physics metadata on a file-by-file basis will be used, optionally fitted with a command line interface to provision files.

3. Providing first time users with instructions and guidance for using Grid based analysis tools. Working examples of Grid analysis sessions, complete with scripts and user algorithms, will need to be supplied.
4. Data storage capacities at each site will need to be clearly defined, with specifications regarding data types and access policy made clearly available to users and production managers.
5. Job submission tools with minimal smarts will be developed, using highly extensible frameworks. This will at first be Grappa (or its equivalent), used as remote job submission interface. Minimal scheduling smarts will be added once the basic submission infrastructure is in place. For example, components to identify where the reconstruction input file collections are located, and components which co-allocate CPU resources. A possible solution involves layering on top of DAGman.
6. Coherent monitoring for the system as a whole will need to be developed:
 - Components which gather and parse Condor log files
 - GridView
 - Real-time network monitoring with graphical display

Metrics for success will include working demonstrations of analysis tasks which produce physics histograms from large collections of DC data, providing much feedback about event throughput, performance, and status throughout the process. There should be opportunity for much user feedback, as users coming into the system with the motivation to extract physics plots will likely be quite vocal (and helpful) as they experience the Grid for the first time.

12.1.3 Challenge Problem II: Athena Virtual Data Demonstration

The following demonstration will be implemented using software developed within the Adagio effort. We will also attempt to employ existing virtual data infrastructure, such as VDL and VDC as developed by CMS and LIGO GriPhyN teams.

We define a query to be an Athena-based consumer of an ATLAS Monte Carlo data (such as from ATLFast) along with a tag that identifies the input dataset needed. In an environment in which user Algorithms are already available in local shared libraries, this may simply be a JobOptions file, where one of the JobOptions (like event selection criteria) is allowed to vary.

Three possibilities will be supported by GriPhyN virtual data infrastructure as implemented with core Athena code supported by VDL, VDC and the Adagio set of extensions to Athena:

1. The dataset exists as a file or files in some place directly accessible to the site where the consuming program will run. In this case, the Athena service that is talking to GriPhyN components (e.g., an EventSelector) will be pointed to the appropriate file(s).

2. The data set exists in some place remote to the executable. The data will be transferred to a directly accessible site, after which processing will proceed as in 1. This is virtual data transparency with respect to location.
3. The data set must be generated. In this case, a recipe to produce the data is invoked. This may simply be a script that takes the dataset selection tag as input, sets JobOptions based on that tag, and runs an Athena-based ATLFAST simulation to produce the data. Once the dataset is produced, processing continues as in 1. This is virtual data with respect to materialization (existence).

Metrics for success will include demonstrated executions for each of the possibilities outlined above, with a verification/monitoring algorithms used to certify results based on pre-calculated sets of histograms.

12.1.4 Demonstrations for ATLAS Software Weeks

A preliminary demonstration of Grappa functionality is planned to be in place for the World-Wide Computing Session of the first ATLAS Software Week in March, 2002. This should include:

1. Authentication to a personal XCAT portal
2. Design of Athena Monte Carlo generation analysis session
3. Selection of several grid resources from the U.S. Testbed, including Condor resources accessed through the Globus Job Manager.
4. Automatic generation of random number seeds for individual jobs
5. Automatic physical file name generation
6. Display of execution monitoring data
7. Preliminary interface displays to Magda metadata and physical replicas for data stored in the Testbed Grid.
8. Demonstration of GridView description, monitoring of Testbed Grid resources

Metrics for success will include real-time demonstration of the Athena analysis chain for user analysis, resulting in displays of physics plots and event throughput monitoring / statistical information during the demonstration.

12.1.5 Demonstration for SC2002

SC2002 will be held in Baltimore, November 16-22, 2002. Demonstration of Grid-based data analysis using ATLAS software and a significant number of Grid sites, beginning first with Tiers 0-2, later expanding to ATLAS Tier 3 sites, and later to non-ATLAS sites such as other sites in the iVDGL. To include:

1. Full chain production and analysis of ATLAS Monte Carlo event data
2. Illustration of typical physicist analysis sessions
3. Graphical monitoring display of event throughput throughout the Grid
4. Live update display of distributed histogram population from Athena
5. Illustration of Challenge Problem I, analysis of DC1 data collections

6. Illustration of Challenge Problem II, virtual data re-materialization from Athena
7. Illustration of Grappa job submission and monitoring examples.

Metrics for success will include a working demonstration which meets the above listed functionality requirements.

12.2 ATLAS Year 3 (October 02 – September 03)

12.2.1 Goals Summary

The first major goal of ATLAS Data Challenge 2 (January 2003 to September 2003) is to evaluate variations to the LHC Computing Model, as currently be debated within the international ATLAS World Wide Computing Group, which is overseen by the National Computing Board (NCB).

During DC2, we will compare a “strict Tier” model in which full copies of ESD data (Event Summary Data) reside on massive tape storage systems and disk at each ATLAS Tier 1 site, to a “cloud” model where the full ESD is shared among multiple sites. The latter results in a complete sample of ESD data on disk at any time. The two models may imply vastly different analysis access patterns, and could result in significant re-direction of facilities resources from computing cycles to network bandwidth capacity, for example. MONARC studies of the new models will be helpful, but the DC will provide the empirical experience from which to complete the design of the LHC computing infrastructure, leading up to the LHC turn-on.

The second major goal of Year 3 is to push development of virtual data technologies for ATLAS, building on the early successes within GriPhyN research on VDL and VDC for CMS and LIGO. At this time the ATLAS core software will be better suited for this type of work. Also needed are tools to evaluate the virtual data reconstruction methods, and algorithms to evaluate their success.

12.2.2 Challenge Problem III: Grid Based Data Challenge

DC2 will use Grid middleware in a production exercise scaled at 10% of the final system. CP-III will require large scale, robust Grid production and analysis tools for data management, job management on distributed resources, security and monitoring.

12.2.3 Challenge Problem IV: Virtual Data Tracking and Recreation

The goal of CP-IV will be the development and demonstration of virtual data re-creation, that is, the ability to rematerialize data from a query using a virtual data language and catalog. Some issues to be resolved:

1. Identify which parameters need tracking to specify re-materialization (things making up the data signature such as code release, platform and compiler dependencies, external packages, input data files, user and/or production cuts).
2. Identify a metric for evaluating the success of re-materialization. For example, what

constitutes a successful reproduction of data products? Assuming bit-by-bit comparison of identical results is impractical, what other criteria can be identified which indicate “good enough” reconstruction? For example, statistical confidence levels obtained by comparison of materialized histograms with reference versions would provide statistics-based criteria for success.

12.3 Overview of Major Grid Goals

Here we list schedules for some of the major GriPhyN goals (GG), challenge problems (CP), in relation to PPDG (PG) and to ATLAS data challenges (DC).

• June 01 - July 02	PG1	Development of Grid based data management with Magda
• Oct 01 – March 02	GG-T2	VDT 1.0 deployment and basic infrastructure
• Dec 01 – Feb 02	GG-T3	Integration of CERN testbed node into US ATLAS testbed
• Jan 02 – July 02	DC1	Data creation, use of Magda, Tier 0-2
• July 02 – June 03	PG2	Job management, Grid job submission
• July 02 – Dec 02	CP-I	Serving data from DC1 to universities, simple Grid job sub.
• Dec 02 – Sept 03	DC2	Grid resource mgmt, data usage, smarter scheduling
• Dec 02 – Sept 03	CP-IV	Dataset re-creation, metadata, advanced data Grid tools
• July 03 – June 04	PG3	Smart job submission, resource usage

Table 12 ATLAS - GriPhyN and PPDG Schedules

	2001				2002				2003				2004		
PG1															
GG-T2															
GG-T3															
DC1															
PG2															
CP-I															
DC2															
CP-IV															
PG3															
Data Management															
Scheduling															

13 Project Management

GriPhyN software development activity, as it pertains to ATLAS, has components in both Software and Facilities subprojects within the U.S. ATLAS Software and Computing Project.

The Level 1 manager of the U.S. ATLAS S&C project is John Huth of Harvard University, also a member of the GriPhyN – ATLAS team. The Level 2 project manager for Software is Torre Wenaus of Brookhaven National Laboratory, who is the ATLAS PPDG project lead and is collaborating with GriPhyN. The Level 2 project manager for Facilities is Rich Baker of BNL who also collaborates with GriPhyN; Rich also supervises Dantong Yu, who coordinates monitoring activities within U.S. ATLAS. The Level 3 project manager for Distributed IT Infrastructure is Rob Gardner of Indiana University, also the project lead for GriPhyN – ATLAS. A Project Management Plan describes the organization of the U.S. ATLAS S&C project. Liaison personnel for GriPhyN have been named for Computer Science (Jennifer Schopf, Globus team) and Physics (Rob Gardner).

13.1 Liaison

Within the U.S. ATLAS S&C project, liaison duties are referenced in Grid WBS 1.3.2 (liaison between U.S. ATLAS software and external distributed computing software efforts). The work items entailed in these roles include:

1. Presentations at various computing reviews (EAC, DOE/NSF, etc.) by appropriate liaison
2. Coordination between U.S. ATLAS S&C personnel and others within the GriPhyN Collaboration
3. Planning, organization of GriPhyN project goals specific to ATLAS

13.2 Project Reporting

Monthly reports will be submitted to the GriPhyN project management. Periodic reviews will be made by the GriPhyN EAC and by the U.S. LHC Project Office. In addition, annual reports will be generated which will give an accounting of progress on project milestones and deliverables. Additional reports, such as conference proceedings and demonstration articles, will be filed with the GriPhyN document server.

14 References

¹ ATLAS: A Torroidal LHC Apparatus, homepage: <http://atlas.web.cern.ch/Atlas/>

² The Athena architecture homepage:
<http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General/index.html>

³ The Gaudi Project: <http://proj-gaudi.web.cern.ch/proj-gaudi/>

⁴ “ATLAS Detector and Physics Performance”, Technical Design Report (ATLAS Collaboration), LHCC 99-14/15, May 1999.
<http://atlasinfo.cern.ch/Atlas/GROUPS/PHYSICS/TDR/access.html>

⁵ U.S. ATLAS Grid Planning page:

<http://ATLASsw1.phy.bnl.gov/Planning/usGridPlanning.html>

⁶ The XCAT Science Portal, Sriram Krishnan, et. al., in proceedings of Supercomputing 2001.

⁷ Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications, D. Gannon, et. al, to appear in the IEEE Journal on Cluster Computing, Special Issue on HPDC01.

⁸ Grappa: Grid Access Portal for Physics Experiments:

- ❑ Homepage: <http://lexus.physics.indiana.edu/griphyn/Grappa/index.html>

- ❑ Scenario document <http://lexus.physics.indiana.edu/~griphyn/Grappa/Scenario1.html>

⁹ CCA: Common Component Architecture forum: <http://www.cca-forum.org/> ;

At Indiana University: <http://www.extreme.indiana.edu/ccat/>

¹⁰ Algorithmic Virtual Data (NOVA project), at BNL:

<http://ATLASsw1.phy.bnl.gov/cgi-bin/nova-ATLAS/clientJob.pl>

¹¹ Condor Project: <http://www.cs.wisc.edu/condor/>

¹² IEPM Group at SLAC: <http://www-iepm.slac.stanford.edu/>

¹³ Abilene Network Engineering team: <http://www.abilene.iu.edu/index.cgi?page=engineering>

¹⁴ Netlogger: A Methodology for Monitoring and Analysis of Distributed Systems, National Energy Research Scientific Computing Center, Computing Sciences Division, Lawrence Berkeley National Laboratory.

<http://www-didc.lbl.gov/NetLogger/>

¹⁵ NPACI Rocks Clusters homepage <http://slic01.sdsc.edu/>

¹⁶ Community Authorization Service (CAS): <http://www.globus.org/security/CAS/>

¹⁷ Workshop to develop an ATLAS – GriPhyN Testbed, Indiana University, June 2000:

http://lexus.physics.indiana.edu/~rwg/griphyn/june00_workshop.html

¹⁸ U.S. ATLAS Testbed workshop, University of Michigan, June 2001: See links from

<http://www.usatlas.bnl.gov/computing/grid/>

¹⁹ GRIPE: Grid Registration Infrastructure for Physics Experiments:

<http://iuATLAS.physics.indiana.edu/griphyn/GRIPE.jsp>

²⁰ GriPhyN Outreach Center: <http://www.aei-potsdam.mpg.de/~manuela/GridWeb/main.html>



GriPhyN Year 2 - CMS Project Plan

21 December 2001

Developed by members of the GriPhyN Project Team

Editors: Mike Wilde and Rick Cavanaugh
wilde@mcs.anl.gov, cavanaugh@phys.ufl.edu

1 Introduction

The CMS GriPhyN efforts during Year 1 consisted primarily of researching the grid requirements for CMS [GriPhyN 2001-1], [GriPhyN 2001-16] and in studying the current status of CMS software, as it relates to the production of simulated CMS data and the future analysis of that data. Such research, which has proven vital to the planning of tasks, will continue in Year 2 as CMS refines and redefines its software environment.

One important consideration is that the immediate CMS demand for software and grid functionality outweighs the current need for scalability. As a result, this project plan details CMS activities for GriPhyN project Year 2, in which we plan to integrate virtual data GriPhyN research results and functionality into the production of CMS simulated data and begin to experimenting with applying virtual data concepts to the problem of CMS data analysis.

1.1 Pertinent CMS Software

While the CMS software is continuing to develop at a rapid pace, it is already highly functional. Indeed, successfully integrating virtual data technologies into an already advanced software environment presents a particular challenge to the CMS GriPhyN team. As there are several packages with which to contend, a brief introduction to some of the pertinent CMS software is given here:

- Simulation of physics events and the CMS detector: Loose collections of Fortran applications, collectively known as CMKIN, simulate different types of physics events resulting from proton-proton collisions. Subsequently, a single Fortran application, CMSIM, simulates the "raw" response of the CMS detector to the physics events produced from the CMKIN application. A typical simulation of 500 events requires approximately 8 hours of processor time (on 1 GHz machine) and produces a 50 GB flat file.
- Reconstruction of physics events: An OO framework, ORCA/CARF, is currently used to reconstruct the "raw" information contained in a physics event and translate it into data, which is

useful for physicists to analyze. The resulting data is stored in an Objectivity database. The input to this reconstruction framework can either be real data as taken from the actual CMS detector or, a simulated data file as produced by the CMKIN/CMSIM applications.

- Mass production of simulated CMS data: A major goal of CMS is the production of large sets of simulated data. The current production of simulated data follows a simple pipeline: (1) input parameters are given to CMKIN/CMSIM which produce a "raw" simulated data file, (2) this data file is copied from a flat file format to an Objectivity database using ORCA/CARF, and finally (3) ORCA/CARF is used to transform the "raw" data into "reconstructed" data and store it in an Objectivity database. IMPALA/BOSS consists of a set of shell scripts, which facilitate the mass production of simulated CMS data by performing input parameter bookkeeping, as well as job submission and tracking. To date, CMS has produced approximately 20 million simulated data events using various versions of IMPALA/BOSS.

This brief CMS software summary will change during Year 2 as the CMS Core Application Software team continues the development process. However, the basic structure of the simulated data production chain is expected to remain relatively constant until sometime in Year 3, when the current Fortran based simulation of the CMS detector will be replaced by an OO based simulation.

1.2 Outline

This document is outlined as follows. First, a broad overview of the GriPhyN activities in CMS is given in Section 2. Section 3 is devoted to a more detailed description of the development tasks foreseen for Year 2 in support of the high-level goals of Section 2. This is followed by a discussion of test bed development in Section 4. A brief conclusion is given in Section 5. Finally, a detailed work plan for Year 2 is presented in the Appendix. The provided work plan is, nevertheless, subject to change as the year evolves.

2 Project Overview

This section presents a broad overview of CMS GriPhyN. It is important to understand that new, and perhaps unexpected, developments will occur during the course of the year. Hence, a substantial amount of time will be budgeted to monitor the progress of CMS software and toolkit development as well as emerging technology from other grid projects and to adapt the CMS GriPhyN Year 2 activities accordingly.

2.1 Goals

The currently identified high-level goals for CMS GriPhyN for year 2 are:

- Show the utility of GriPhyN technology as a basis for enhancing the robustness and reproducibility of distributed computing, by integrating grid components more deeply into the CMS production software, with the results of the integration being used either in real production or in challenge demos.
- Gain and demonstrate the commitment of CMS to the evaluation, testing, integration, and use of GriPhyN technologies.
- Create a test bed in which both GriPhyN and PPDG CMS activities can be conducted.
- Forge joint activities and tighter coordination between PPDG, European Data Grid, TeraGrid, and GriPhyN (as CMS is part of all four grid projects).
- Apply GriPhyN virtual data research to CMS simulation production. Deploy the virtual data mechanism and use it to provide automated production as well as a GriPhyN laboratory.
- Integrate the virtual data catalog into an important production application, and demonstrate the benefits of detailed data derivation tracking and large-scale data re-derivation
- Instrument and measure the use of virtual data and request planning mechanisms to gather data and feedback for further CS research.
- Apply preliminary GriPhyN research results and execution planning and scheduling mechanisms (for example, using DAGMan) to CMS production.

<u>Name</u>	<u>Affiliations</u>	<u>Role</u>
James Amundson	PPDG FNAL CMS	CMS Physicist; MOP Developer
Paul Avery	GriPhyN UFL CMS	GriPhyN PI; UFL CMS Physics Lead
Lothar Bauerdick	GriPhyN FNAL CMS	US CMS Computing Coordinator
Dimitri Bourilkov	GriPhyN UFL CMS	CMS Physicist
Rick Cavanaugh	GriPhyN UFL CMS	CMS Physicist
Greg Graham	FNAL CMS	CMS Physicist; IMPALA Developer
Koen Holtman	GriPhyN CIT CMS	CMS Computer Scientist
Iosif Legrand	PPDG CIT-CERN-CMS	CMS Physicist
Vladimir Litvin	GriPhyN CIT-CMS	CMS Physicist
Harvey Newman	GriPhyN PPDG CIT-CMS	CMS Physicist; CMS GriPhyN Lead
Rajesh Rajamani	Condor UW-CS	Computer Scientist, CMS Application Support
Jorge Rodriguez	GriPhyN UFL-CMS	CMS Physicist
Conrad Steenberg	PPDG CIT-CMS	CMS Physicist
Jens Voeckler	GriPhyN UC-CMS	Computer Scientist
Edwin Soedermadji	iVDGL CIT-CMS	CMS Computer Scientist
Suresh Singh	GriPhyN CIT-CMS	CMS Computer Scientist
Julian Bunn	GriPhyN CIT-CMS	CMS Physicist
Takako Hickey	PPDG CIT-CMS	CMS Computer Scientist

Table 1. CMS GriPhyN Team. This table lists the people expected to contribute to CMS work in GriPhyN over Year 2. While many people are directly associated with GriPhyN, several are affiliated with other grid projects, or with CMS. This inter-project team make-up is considered important for the integration and use of tools developed outside of GriPhyN. It is hoped to expand this collaboration to include members of the European DataGrid and the CMS DataGrid.

- Start exploring the CMS analysis process, creating prototypes of GriPhyN-based analysis systems that can lead the way to live science use in project year 3.

2.2 Activities

Personnel involved in the CMS GriPhyN activities in Year 2 are listed in Table 1. The activities for this team during Year 2 will consist of test bed development, integration of VDT technology into CMS simulation production, and the prototyping of using VDT technology in the CMS analysis process. These are described in the sections below.

2.2.1 Production of CMS Simulated Data

The main GriPhyN CMS effort for Year 2 is to integrate virtual data and request-planning mechanisms into the IMPALA/BOSS production system of CMS simulated data. As part of this effort, several job management mechanisms need to be brought together into a single coherent entity. This will be accomplished by extending IMPALA/BOSS with GriPhyN VDT components as well as prototype virtual data technologies. Valuable research feedback to the GriPhyN project is anticipated as VDT enhanced versions of IMPALA/BOSS are evaluated by CMS. During Year 2 and beyond, we intend to integrate several GriPhyN Grid technologies into the IMPALA/BOSS job management, including: replication, replica location service, reliable file transfer, the Community Authorization Service, and prototypes of the GriPhyN Job Execution Planner.

The benefits to CMS of this effort represent increased functionality: easier, more automated job submission; easier recalculation of derived data products; accurate tracking of data derivation; ability to re-derive latter stage outputs of the production data pipeline without re-calculating the earlier phases, in cases where the later-stage processing programs require changes. We intend to highlight ease of grid usage as a major benefit to CMS, especially in the automated handling of failures and complex grid configuration and usage details. We also intend to explore how to effectively utilize the GDMP publish-subscribe paradigm within CMS production.

The benefits to GriPhyN of this effort include: live testing of a fundamental GriPhyN concept in intensive production for real users; user feedback on the value of the virtual data paradigm and usability of the tool

set; measurement of virtual data process effectiveness; capture of live logs of the detailed activity and resource utilizations of the production process for further CS research.

2.2.2 Analysis of CMS Data

This activity explores the vital later-phase of CMS data analysis. Such analyses search massive numbers of events for signature patterns offering evidence for various proposed theories of nature. Once CMS is online and recording live data from the detector, analysis activities will be the dominant use of computing resources.

In a typical analysis, a physicist selects events from a large database consisting of small sized event classification information, or TAG data. Using the TAG data, the physicist gathers the full reconstructed event from various sources including, if necessary the creation of fully reconstructed versions of those events if they do not already exist on some "convenient" storage system. A typical analysis might look something like the following:

- Search through 10^9 TAG events (~ 100 GB), select 10^6 of those events and fetch the full "raw" event data for the selected events (~ 1 TB)
- Invoke a user defined reconstruction of the full "raw" event data and make a user defined set of Analysis Object Data (~ 100 GB) and TAGs (~ 100 MB)
- Analyze the user defined Analysis Object Data (AOD) and TAG datasets interactively, extracting a few hundred candidate signal events (~ 10 MB).
- Histogram the results and visualize some of the "interesting" events.

We seek to create virtual data techniques to track data dependencies for the files and/or objects in this process from TAG schemas and TAG databases (or tables) back to the reconstructed event sets and possibly back to the raw data.

We will build tools for this type of user-driven fine-granularity physics dataset extraction and transport over the grid, driven by an easy-to-understand interface that reduces the difficulties of marshalling distributed grid resources. This effort will exploit newer collection management features developed by ORCA/CARF team at CERN.

We expect to further explore the impact of workloads from end-user physics data analysis on the grid system, by prototyping distributed end user analysis tools, demos, and pilot facilities.

2.2.3 Test Beds

To facilitate the research, development, and integration of virtual data technology into the CMS software environment, several potential grids may be used for CMS GriPhyN Year 2 activities:

- GriPhyN Test Grid: an envisaged, shared GriPhyN experimental grid meant to be used by all experiments, for the initial stages of prototyping.
- US-CMS Test Grid: a CMS-only, experimental test bed (currently under construction), to be used by CMS GriPhyN and CMS PPDG collaborators to prototype tools for CMS production of simulated data and distributed analysis.
- CMS Data Grid: a CMS-only test bed (currently under construction) to be used by CMS collaborators to prototype tools for CMS production of simulated data and distributed analysis.

One of the early goals will be to integrate the US-CMS Test Grid with the CMS Data Grid. This will allow the CMS GriPhyN team to remain synchronized and integrated with future CMS developments.

3 Development Tasks

The primary focus of CMS activities in GriPhyN during Year 2 will be the development of virtual data tools for the production of Monte Carlo simulated CMS data and virtual data tools for the analysis of CMS data. These software packages will rely both upon existing tools from the current Virtual Data Toolkit as well as contribute new tools, which are general enough in nature, into future versions of the Virtual Data Toolkit.

In support of these activities, efforts will be directed towards the establishment of a catalog infrastructure including: Replica Catalogs (RC), Virtual Data Catalogs (VDC), and Meta-Data Catalogs (MDC). Significant progress towards the development of a prototype VDC has already been accomplished by Voeckler using a PostgreSQL database coupled with a Perl interface. The catalog tracks the dependencies of data files and transformations between data files. As such, it is able to regenerate any (missing or deleted) derived data file on demand.

While CMS does not currently use virtual data concepts, CMS has detailed several future needs related to virtual data [GRIPHYN 2001-16]. Using the experience gained by integrating the VDC with current CMS production and analysis needs (see below), CMS-GriPhyN plans to work with CS-GriPhyN to further develop virtual data concepts by taking the following as work items during Year 2:

- Work Item 1: Develop and prototype the concept of "grid uploaded" files and algorithms (i.e. transformations between data files) by leveraging existing technology in GDMP and extending it. Such files and transformations would exist in a grid wide replica catalog and be distinguished by Unique Identifiers (UID).
- Work Item 2: Interface the current VDC with GDMP and an extended GDMP so that virtual data products can be materialized from "grid uploaded" files and transformations. Each materialized virtual data product would receive a UID and entry into the "replica catalog" and/or the VDC. Platform dependencies and their relation to virtual data product UIDs will be investigated.
- Work Item 3: Develop prototypes for Consistency Management of the replica catalog over a grid.

3.1.1 Tools for the Production of Monte Carlo Simulated CMS Data

A tool for the distributed production of CMS Monte Carlo simulated data, known as MOP, is currently under development from the Particle Physics Data Grid. MOP (which is based upon Globus, Condor-G, and DAGMan) is loosely integrated with a set of shell scripts, known as IMPALA, that are currently used by CMS for Monte Carlo production as well as the Grid Data Movement Package, or GDMP. Currently these tools do not employ virtual data concepts.

Over the course of Year 2, CMS-GriPhyN will augment MOP and IMPALA with virtual data tracking using the Virtual Data Catalog. This will involve decomposing the job submission and bookkeeping logic of IMPALA into parameters and transformations which are specific to CMS and logic which is more general to batch job execution planning in the form of abstract Directed Acyclic Graphs (DAGs). In addition, the distributed job execution logic of MOP will be embedded into the VDC to facilitate virtual data materialization in a grid environment. In order to fine tune these concepts and synchronize with CMS production efforts, two challenge problems are proposed over the next year:

- Challenge Problem 1: Produce 50,000 Monte Carlo fully simulated CMS events (including pileup) using the VDC on a USCMS Test Grid (see below). This should expose any technical and conceptual modifications required to use the VDC in realistic situations.
- Challenge Problem 2: Fulfill one (or several) official Monte Carlo Production request(s) from CERN on a USCMS Test Grid. This will demonstrate the feasibility of using the VDC in "real world" CMS production activities.

The aim of this effort is two-fold: 1) provide an ever more autonomous environment for CMS Monte Carlo production by enabling automatic error recovery, rigorous bookkeeping, and transparent production at different CMS grid sites and 2) provide valuable insight into virtual data concepts for future prototyping.

3.1.2 Prototypes for the Analysis of CMS Data in a Remote Environment

Virtual data as it applies to scientific analysis of CMS data has only recently been considered. It is currently unclear whether CMS physicists will employ a single monolithic analysis tool, or use a standardized set of analysis tools, or even use different sets of analysis tools. As a result, CMS research into different data analysis paradigms will continue to be monitored by CMS-GriPhyN throughout Year 2.

Given that CMS requires that physicists have the option of defining their own sets of data products (files, objects, etc) for scientific analysis, it is important to begin the process of tracking virtual data products as they relate to end-user data analysis. In order to probe this and to facilitate whatever analysis tool(s) that

may be used in the future by CMS, a remote data server (known as Clarens) is being developed by Steenberg to enable analysis of CMS data distributed over a wide-area network. Clarens is based on a Client/Server approach and provides a framework for remote data analysis. Communication between the client and server is conducted via XML-RPC over a scalable SSL-encrypted HTTP transport provided by the Apache web server. Currently the server is only linked to the standard CMS analysis C++ libraries, but various server functions are envisioned. The client can be end-user specific and implementations currently exist for several data analysis tools including: C++, Python (for use in the Lizard analysis environment), PHP (a web portal), The Java Analysis Studio, and a Python plug-in for SciGraphica. This allows the user full access to remote CMS data via a choice of analysis packages as well as the web.

During Year 2, Steenberg plans to grid-enable Clarens by taking full advantage of the Virtual Data Toolkit including: the Virtual Data Catalog for tracking CMS data, the Globus Security Infrastructure for authentication, and Grid-ftp for CMS data movement:

- Work Item 4: Provide a remote interface to a useful subset of VDT 1.0 functionality, including authentication, file movement, and job scheduling and monitoring.
- Work Item 5: Provide a remote interface to the VDC.

As a joint endeavor with efforts in the virtual data tracking of CMS Monte Carlo production, the following data challenge problem is proposed:

- Challenge Problem 3: Remotely analyze 50,000 events using Clarens integrated with the VDC as used in Challenge Problem 1.

Finally, investigations into more fine grained data collections at the object level and their relation to a VDC will also be done during Year 2.

4 Infrastructure Tasks

CMS GriPhyN is currently building a US-CMS Test Grid, in cooperation with the Particle Physics Data Grid, at five initial sites: The California Institute of Technology, Fermi National Accelerator Laboratory, The University of California-San Diego, The University of Florida, and The University of Wisconsin-Madison. The US-CMS Test Grid is expected to be operational in January 2002.

The initial goals of the test grid will be to produce a platform, which enables CMS grid software development, and which probes policy issues related to Certificate Authorities. In addition, early integration with the CMS Data Grid (also expected to come online in January 2002) will be aggressively pursued. This will facilitate a closer technical cooperation between CMS and GriPhyN.

4.1.1 Software Components

To ensure interoperability with other GriPhyN grid efforts, the US-CMS Test Grid is based upon the GriPhyN Virtual Data Toolkit Version 1.0, which includes Condor 6.3.1, Globus 2.0 beta, and GDMP 2.0. In addition to the packages contained in the Virtual Data Toolkit, Condor-G 6.3.1 and Objectivity/DB will be installed.

- Work Item 6: Install and verify the Virtual Data Toolkit 1.0 and Condor-G 6.3.1 at each grid site.
- Work Item 7: Install Objectivity 6.1 at each grid site

The entire suite of CMS software is complex with many external software package dependencies and dynamically linked shared object libraries. As a result, a CMS software Distribution After Release (DAR) package has been developed at Fermilab. DAR files are essentially "tar balls" consisting of particular versions of CMS executables packaged with the required CMS execution environment (shared object libraries, etc). It is planned to distribute and install DAR files to each grid site, enabling grid users to have access to several versions of CMS software.

- Work Item 8: Install an initially agreed upon CMS DAR file at each grid site

To facilitate sophisticated CMS software development, dynamic installation of personalized CMS software will be investigated and implemented via DAR files.

- Work Item 9: Investigate "on-the-fly" DAR file installation via a remote user.

4.1.2 Platform Requirements

Currently, the CMS software environment supports the Red Hat 6.1, 6.2 and Solaris operating systems. To ensure compatibility with the CMS software environment as well as the VDT 1.0, the test bed will be entirely composed of machines running Red Hat 6.2.

4.1.3 Security and Resource Sharing Policies

Each CMS-GriPhyN and CMS-PPDG registered user will receive an account at each of the five grid sites. Currently, the account management for the US-CMS Test Grid is de-centralized. During the course of Year 2, the CMS GriPhyN team will monitor developments in the European Data Grid Project regarding account management.

The test bed will initially use the Globus Certificate Authority for the distribution of user and gatekeeper authentication certificates. However, as ESnet is expected to provide a Certificate Authority in April 2002, the US-CMS Test Grid will migrate to the ESnet Certificate Authority when it becomes available.

- Work Item 10: Monitor the progress of the ESnet Certificate Authority and migrate to ESnet issued GSI authentication certificates as they becomes available

A mechanism for centralized accounting of resource sharing and authorization has not been addressed. The CMS GriPhyN team will monitor progress with the Community Authorization Service (CAS), currently being developed by Globus. However, CAS is considered a long-term solution. In the absence of such a mechanism, resource sharing and accounting procedures will be studied and implemented on a site-by-site basis.

4.1.4 Integration with other Grid Projects

The integration of the US-CMS Test Grid with other Grid Projects of the same scope is considered important. In particular, preliminary dialog has already begun with the CMS Grid Integration Task Group concerning how best to integrate the CMS Data Grid and the US-CMS Test Grid. This will provide a higher likelihood that tools developed by CMS GriPhyN will more directly benefit and be accepted by CMS.

- Work Item 11: Determine and resolve any possible conflicts between the US-CMS Test Grid and the CMS Data Grid.

5 Conclusions

The CMS GriPhyN activities in Year 2 will involve three primary research areas: (1) development and integration of virtual data technology into the production of CMS simulated data, (2) research and development of virtual data functionality in remote analysis of CMS data, and (3) development of the necessary test bed infrastructure needed to develop, test, and integrate virtual data into the CMS grid environment.

A great deal of effort has already been invested in understanding the current and future virtual data needs of CMS. This investigative effort will continue during Year 2 so that GriPhyN and CMS remain synchronized with respect to milestones and software architecture. Nevertheless, additional liaison efforts between CMS and GriPhyN are required to ensure that virtual data technologies developed by GriPhyN are not only found to be useful in CMS, but actually integrated into CMS grid technology as well. The integration of the US-CMS Test Grid and the CMS Data Grid will assist this effort.

6 Appendix: Snap-shot of current Year 2 Work Plan (subject to change)

- 1 CS work on Distributed Virtual Data Catalogs
 - 1.1 Develop Replica Catalog
 - 1.1.1 Investigate concepts of "uploaded" files and algorithms
 - 1.1.2 Prototype a Grid-wide RC which catalogs files and algorithms
 - 1.1.3 Investigate Consistency Management Techniques with respect to file existence
 - 1.2 Integrate the VDC/L with the RC
 - 1.3 Integrate VDC with mop_submitter
- 2 Virtualize CMS Distributed Monte Carlo Production
 - 2.1 Integrate CMS Monte Carlo Production Pipeline with the VDC/L
 - 2.1.1 Investigate "on-the-fly" creation of Objectivity database files and how to catalog them in the VDC
 - 2.1.2 Integrate VDC directly with DAR (tar ball of the CMS software and environment)
 - 2.1.2.1 Understand the architecture of DAR
 - 2.1.2.2 Produce a VDL file for a DAR pipeline
 - 2.1.3 Include Pile-up in VDL file for Monte Carlo Production
 - 2.1.4 Start a test Monte Carlo Production using the VDC with no validation checks
 - 2.1.5 Produce 10 000 events using the VDC
 - 2.1.6 Validation of Monte Carlo Simulated Data
 - 2.1.6.1 Understand how to validate a produced data file
 - 2.1.6.2 Determine additional steps in the production pipeline to validate each produced data file
 - 2.1.6.3 Integrate additional validation steps into the production pipeline
 - 2.1.7 Start actual Monte Carlo Production using the VDC with automatic validation checks
 - 2.1.8 Produce 50 000 events using the VDC
 - 2.2 Develop VAMPALLA: Virtual dAta iMPALLA
 - 2.2.1 Develop Test Version of a Virtual Data IMPALA
 - 2.2.1.1 Understand the architecture of IMPALA
 - 2.2.1.2 Understand the architecture of MC_Runjob
 - 2.2.1.3 Understand the architecture of BOSS
 - 2.2.1.4 Determine in which initial layer the VDC should be integrated
 - 2.2.1.5 Write interface between the IMPALA integration Layer and the VDC
 - 2.2.1.6 Deploy Test version of VD-IMPALA for Monte Carlo Production
 - 2.2.1.7 Receive Comments
 - 2.2.1.8 Determine a more optimal architecture for a Virtual Data IMPALA

- 2.2.2 Virtual Data IMPALA Version
- 2.2.2.1 Virtual Data IMPALA Version 1
- 2.2.2.2 Virtual Data IMPALA Version 2
- 2.2.2.3 Virtual Data IMPALA Version 3
- 2.2.2.4 Virtual Data IMPALA Version 4
- 2.2.2.5 Virtual Data IMPALA Version 5
- 2.2.2.6 Virtual Data IMPALA Version 6
- 2.2.2.7 Virtual Data IMPALA Version 7
- 2.2.2.8 Virtual Data IMPALA Version 8
- 2.2.2.9 Virtual Data IMPALA Version 9
- 2.2.2.10 Virtual Data IMPALA Version 10
- 2.2.2.11 Virtual Data IMPALA Version 11
- 2.2.2.12 Virtual Data IMPALA Version 12
- 2.2.2.13 Virtual Data IMPALA Version 13
- 3 Virtualize CMS Remote Data Analysis
- 3.1 Integrate Clarens with VDT 1.0
- 3.1.1 Authenticate connection using GSI
- 3.1.2 Demonstrate GSI authentication at GriPhyN CS-meeting
- 3.1.3 Expose the GDMP C++ API to remote clients
- 3.1.4 Demonstrate text version of GDMP C++ API to remote clients
- 3.1.5 Demonstrate GUI version of GDMP C++ API to remote clients
- 3.1.6 Expose the Globus Toolkit API to remote clients
- 3.1.7 Demonstrate text version of Globus API to remote clients
- 3.2 Integrate Clarens with VDC/L
- 3.3 Remotely analyze 50 000 events using Clarens, the VDC and the VDT
- 3.4 Release Clarens 1.0 beta
- 3.5 Receive feedback on Clarens 1.0 beta from users
- 3.6 Interface Clarens with Root
- 3.6.1 Provide a Clarens data server for the Root I/O Package
- 3.6.2 Provide a Clarens data client for the Root Analysis Package
- 4 US CMS Test Beds
- 4.1 Establish Test Grid (based on VDT 1.0 and Globus CA)
- 4.1.1 Agree on an Initial Account Management Policy
- 4.1.2 Agree on Software Base
- 4.1.3 Commission Caltech Site

- 4.1.3.1 Identify Machines for US CMS Test Bed
- 4.1.3.2 Provide initial user accounts
- 4.1.3.3 Deploy Software
 - 4.1.3.3.1 Install VDT 1.0
 - 4.1.3.3.1.1 Condor 6.3.1
 - 4.1.3.3.1.2 ClassAds 0.9
 - 4.1.3.3.1.3 DAGMan 6.3.1
 - 4.1.3.3.1.4 Globus 2.0 beta
 - 4.1.3.3.1.5 GDMP 2.0
 - 4.1.3.3.2 Install Condor-G 6.3.1
 - 4.1.3.3.3 Install Objectivity 6.1
 - 4.1.3.3.4 Install initial version of DAR
- 4.1.4 Commission Fermilab Site
 - 4.1.4.1 Provide initial user accounts
 - 4.1.4.2 Deploy Software
 - 4.1.4.2.1 Deploy VDT 1.0
 - 4.1.4.2.1.1 Condor 6.3.1
 - 4.1.4.2.1.2 ClassAds 0.9
 - 4.1.4.2.1.3 DAGMan 6.3.1
 - 4.1.4.2.1.4 Globus 2.0 beta
 - 4.1.4.2.1.5 GDMP 2.0
 - 4.1.4.2.2 Deploy Condor-G 6.3.1
 - 4.1.4.2.3 Deploy Objectivity 6.1
 - 4.1.4.2.4 Deploy initial version of DAR
- 4.1.5 Commission UCSD Site
 - 4.1.5.1 Provide initial user accounts
 - 4.1.5.2 Deploy Software
 - 4.1.5.2.1 Deploy VDT 1.0
 - 4.1.5.2.1.1 Condor 6.3.1
 - 4.1.5.2.1.2 ClassAds 0.9
 - 4.1.5.2.1.3 DAGMan 6.3.1
 - 4.1.5.2.1.4 Globus 2.0 beta
 - 4.1.5.2.1.5 GDMP 2.0
 - 4.1.5.2.2 Deploy Condor-G 6.3.1
 - 4.1.5.2.3 Deploy Objectivity 6.1

- 4.1.5.2.4 Deploy initial version of DAR
- 4.1.6 Commission Florida Site
 - 4.1.6.1 Provide initial user accounts
 - 4.1.6.2 Deploy Software
 - 4.1.6.2.1 Deploy VDT 1.0
 - 4.1.6.2.1.1 Condor 6.3.1
 - 4.1.6.2.1.2 ClassAds 0.9
 - 4.1.6.2.1.3 DAGMan 6.3.1
 - 4.1.6.2.1.4 Globus 2.0 beta
 - 4.1.6.2.1.5 GDMP 2.0
 - 4.1.6.2.2 Deploy Condor-G 6.3.1
 - 4.1.6.2.3 Deploy Objectivity 6.1
 - 4.1.6.2.4 Deploy initial version of DAR
- 4.1.7 Commission Wisconsin Site
 - 4.1.7.1 Provide initial user accounts
 - 4.1.7.2 Deploy Software
 - 4.1.7.2.1 Deploy VDT 1.0
 - 4.1.7.2.1.1 Condor 6.3.1
 - 4.1.7.2.1.2 ClassAds 0.9
 - 4.1.7.2.1.3 DAGMan 6.3.1
 - 4.1.7.2.1.4 Globus 2.0 beta
 - 4.1.7.2.1.5 GDMP 2.0
 - 4.1.7.2.2 Deploy Condor-G 6.3.1
 - 4.1.7.2.3 Deploy Objectivity 6.1
 - 4.1.7.2.4 Deploy initial version of DAR
- 4.1.8 Verify inter-site Test Grid functionality
- 4.1.9 Deploy version of the SC2001 MOP demo
- 4.1.10 Deploy version of the SC2001 VDC demo
- 4.2 Investigate currently available technologies for centralized New Account Registration
- 4.3 Integrate Test Grid with CMS Data Grid
 - 4.3.1 Determine any infrastructure requirements and resolve possible conflicts
 - 4.3.1.1 Resolve any Certificate Authority issues
 - 4.3.1.2 Agree to use a common version of Globus and Condor
 - 4.3.1.3 Determine if AFS is required as a shared file system
 - 4.3.1.4 Detail any required software

- 4.3.2 Agree on an Account Management Policy
- 4.3.3 Implement any required compatible infrastructure
- 4.3.4 Create user accounts according to agreed policy
- 4.3.5 Verify Inter-grid test bed functionality
- 4.4 Upgrade Test Grid to use the ESnet Certificate Authority
- 4.4.1 ESNet CA comes online
- 4.4.2 Replace the globus-gatekeeper certificate at each site
- 4.4.3 Each user submits a new grid-mapfile entry to each site
- 4.5 Upgrade Test Grid to VDT 2.0
- 4.6 Perform a CERN request for Monte Carlo Production using MOP
- 4.7 Establish Production Grid (based on VDT 2.0 and ESNet CA)
- 4.7.1 Develop US-CMS Production Grid Plan, Version 1
- 4.7.2 Circulate US-CMS Production Grid plan for feed back
- 4.7.3 US-CMS Production Grid Plan approved
- 4.7.4 Allocate machines and/or purchase machines for US-CMS Production Grid
- 4.7.5 Install VDT 1.0 on US-CMS Production Grid machines
- 4.7.6 Verify US-CMS Production Grid functionality
- 4.7.7 US-CMS Production Grid Operational
- 4.8 Commence regular Monte Carlo Production using IMPALA/MOP on US-CMS Production Grid
- 4.9 Perform a CERN request for Monte Carlo Production using a Virtual Data IMPALA on Test Grid
- 4.10 Commence regular Monte Carlo Production using Virtual Data IMPALA



DRAFT: COMMENTS AND MATERIAL SOLICITED

GriPhyN-LIGO Project Plan for Year 2

December 22, 2001

Developed by members of the GriPhyN-LIGO Project Team

Submit changes and material to: Ewa Deelman, editor
deelman@isi.edu

1 Project Year 2: October 1, 2001 to September 30, 2002

In the first year and the first quarter of Year 2, we focused on basic Virtual Data aspects, such as materializing data based on a user's request.

We have constructed a prototype, which was shown at the SC 2001 conference in Denver. This demo exposed much of the GriPhyN infrastructure. A Virtual Data Request was formed by a broker on behalf of the client; this XML document was sent to the Planner, which planned the necessary computations and data movements based on the resources available at Caltech, UWM and the SC showfloor. The plan was then sent to CondorG for execution, computation was initiated and monitored, and the result delivered to the client.

We also demonstrated the feasibility of the integration of Grid middleware, such as Globus and CondorG with the existing LIGO Data Analysis System (LDAS). As part of this integration, Globus services such as the Globus Resource Allocation Manager (GRAM) were interfaced to the LDAS job submission system. Functionality was also provided to enable the staging of data in and out of LDAS via the GridFTP protocol. As a result, the Globus Security Infrastructure was provided as a secure way of accessing the LDAS compute resources at Caltech and UWM.

In the second year of the project, we will focus our efforts in four directions:

- Increase the complexity of Virtual Data requests and increase the amount of available Virtual Data.
- Further investigation of Virtual Data concepts, including the evaluation and implementation of the Transformation Catalog.
- Request planning, investigate available methodology and examine the fault tolerance requirements.
- Evaluate the use of Globus Grid Security Infrastructure (GSI) and the newly developed Community Authorization Service (CAS) in the LIGO environment.

A unifying framework for this work is the Search for Continuous Wave (or Pulsar) Sources of Gravitational Waves which has been posed in the LIGO collaboration. This challenge is focused on large-scale pulsar search and is described in the next section.

2 The Pulsar Search Mock Data Challenge

The Pulsar search takes 1-D time series strain data (which is termed the gravitational wave channel) and creates 2-D frequency-time (f-t) maps or images. These are then processed using digital signal processing techniques to look for evidence of weak, continuous-wave (CW), monochromatic spectral features. The search is conducted in several stages.

First, the gravitational wave channel is extracted from full frame files. This channel is corrected for dropouts, instrumental response function (the calibration of the channel from ADC counts to physical units), and down-sampled from the original 16.384 kHz to 2.048 kHz. Then 1800 1-second, single channel frames are concatenated to form 30 minute duration frames. In order to build a time-frequency image, Short Fourier Transforms (SFTs) are created on the data using code that uses LDAS. Each resulting frame is now a composite of many SFTs that correspond to the original 30 minutes of time series data. These processed frame files are approximately 10 Mbytes in size. The SFT frames can be further combined to construct time-frequency images that refer to about 6 months of data over a narrow frequency range.

This processing requirement fits well into the GriPhyN Virtual Data Grid model. The basic elements of the GriPhyN VDT can be used to materialize these derived data products, based on user requests. For example, if a scientist wishes to search for a known pulsar in the data, he would request a 2-D image spanning the narrow frequency band (bandwidth of a $\sim \text{few} \times 10^{-3}$ Hz.) for a range of time, and build the corresponding virtual data request. This request would then be serviced by the GriPhyN infrastructure.

Although we addressed issues of channel extraction and transpose in the current prototype this Mock Data Challenge brings with it new challenges. We now have a wider range of derived data products, some of them in the frequency domain.

3 Virtual Data Requests

Our goal in Year 2 is to expand on the base system developed in Year 1 by continuing research on scientific Virtual Data for LIGO, extending the scientific complexity and sophistication of the Virtual Data products that GriPhyN can provide, as well as the quantity of data that is accessible by Grid tools. In order to accomplish that, we need to address the following issues:

Grid-enable the scientific analysis of pulsar application codes. Continue the development of the Globus GRAM interface to the LIGO Data Analysis System (LDAS), with an emphasis on registration of virtual data products with the LIGO event monitor database. LIGO data replication to Tier II sites using the Globus Replica Catalog tools and CondorG, along with registration of replicated data and subsequent virtual data (since Tier II sites may post-process raw data before archiving). In order to give the GriPhyN-LIGO Virtual Data (GLVD) access to all LIGO data, it needs to be able to see the catalog of such data. As the LDAS DB2 database becomes synchronized with the contents of the HPSS and LHO/LLO holdings, so the Replica/Derived-data catalogs should mirror it.

Specific milestones are as follows:

Q1: [UWM] Prototype a Globus/LDAS interface, which uses GSI security. (done)

Q2: [UWM] Broaden the GRAM/LDAS interface to accommodate a greater variability and functionality of LDAS Virtual Data Products, such as SFTs, concatenation, decimation and resampling.

Q3: [CIT, USC, UWM] Design a Data Discovery mechanism for discovery of data replicas on a Grid. One focus of this work is to take into account the ability to interact with the LDAS Diskcache resources in order to enable external visibility of LIGO/LDAS data on the Grid.

Q4: [CIT, USC, UWM] Implementation of the Data Discovery mechanism to support the pulsar search.

4 Virtual Data Concepts

So far, we have explored only a very small subset of LIGO's Virtual Data space. In the second year, we plan to expand this further by providing support for a larger number of transformations used to create Virtual Data products. In the first quarter of year 2, we have proposed a design for the Transformation Catalog (TC), which analogous to the Replica Catalog, performs a mapping from a logical space to the physical space. In the TC, logical transformation names are mapped to physical instances of the transformations, indicating the support OS platform, configuration files needed, etc... Initially, we will populate the catalog with the transformations needed to support the pulsar search. We expect that implementing and experimenting with the TC will allow us to better understand Virtual Data concepts.

A pulsar search is a prime candidate for the application of virtual data concepts and grid based computational tools. This is because there are many possible transformations that one can perform on these data, each specified by a putative source location on the celestial sphere and by parameters that characterize source's intrinsic properties. The entire data set for a one-year search might be of order: $(2048 \text{ samples/sec}) \times (2 \text{ bytes/sample}) \times (3 \times 10^7 \text{ sec}) = 120 \text{ Gbytes}$. This sample set covers a 1kHz frequency bandwidth.

A given pulsar will emit in a much narrower frequency range. Frequency variations arise from modulation due to the earth's motion around the sun, resulting in approximately a frequency variation of $df/f \sim 10^{-4}$. Consequently, a year of data can be searched for a 1 kHz pulsar by using only $(10^{-4}) \times 120 \text{ Gbytes} = 12 \text{ Mbytes}$ of data. A reasonable chunk of data to process at once is 50 Mbytes. A priori, though, it is not known *which* 50 Mbyte subset of the data to use.

A simple strategy for grid-enabling a wide-area pulsar search is as follows.

- A server distributes ~ 50 Mbytes of data to a large number of clients
- Each client searches this 50 Mbytes of data for pulsars located anywhere on the sky (or in some specific subregion)
- The client can search in principle about 10^{14} sky locations, so the client can compute "forever" based on this small data volume.
- Each time that a candidate source is found at a given sky location and set of intrinsic source parameters, the relevant parameters are returned to the server.
- Statistical studies and more extensive follow-up analysis are done on the candidate sources. One of the LIGO collaboration groups at the Albert Einstein Institute in Potsdam, Germany will be producing a standalone code that implements this strategy, for use on a dedicated computing cluster.

The immediate task will be migrating and adapting this code to one that can use the entire iVDGL and GriPhyN computing grid.

Milestones:

Q1: [USC] Design the Transformation Catalog. (done)

Q2: [USC] Implement the Transformation Catalog.

Q3: [CIT, USC, UWM] Explore the design of the Derived Data Catalog, which specifies how Virtual Data products are materialized.

Q3: [USC,UC, ANL] Unification of the catalog schemes used by CMS and LIGO – basing it on a common VDT 2.0 release.

Q4: [CIT, UWM] Apply replication concepts by developing a real-time international mirror, and a fault-tolerance replica at UW-Mil. This will help us clarify what are the issues in performing bulk operation in the Virtual Data Grid environment Use of the Transformation Catalog to materialize Virtual data required in the pulsar search.

5 Request Planning and Fault Tolerance

So far, planning has been performed on a very small scale. In year 2, we will evaluate the existing planning techniques and determine their applicability to the GriPhyN planning problem. Our Planner will need to take into account a large number of resources and initially focus on processing single requests. In the future, we will also explore bulk request processing.

Milestones:

Q2: [USC] Specify the planning requirements.

Q3: [USC] Evaluate the available planning software.

Q4: [USC] Prototype a reasonably sophisticated planner for GLVDG.

Another area not addressed yet is the problem of fault tolerance. Here, we need to perform the following tasks:

- Specify fault tolerance requirements for LIGO and for GriPhyN in general.
- Assess existing fault and failure issues within the LIGO application.
- Assess the applicability of existing fault tolerance analysis techniques for distributed systems to GriPhyN/LIGO.
- Explore approaches for producing a fault tolerant DAGMan.

As part of our fault tolerance analysis, we will define a framework for fault tolerance analysis within the context of GriPhyN, at several levels: system, task and DAGMan levels.

We will also perform a design and sensitivity analysis by:

- Exploring fault tolerance issues of alternative designs.
- Exploring sensitivity of alternative designs to various failures.

Finally, we will evaluate techniques for performing testing and diagnostics, with emphasis on fault/failure detection and recovery.

Milestones:

Q2: [USC] Specify fault tolerant requirements. Assess existing fault/failure issues within LIGO.

Q3: [USC] Define framework for fault tolerant analysis. Assess the applicability of existing techniques to GriPhyN/LIGO. Explore approaches for producing a fault tolerant DAGMan.

Q4: [USC] Recommend an approach for the incorporation of fault tolerance in LIGO.

6 Security

LIGO has sophisticated requirements where different groups need to access different parts of the data stream. While the Globus model can deal with this, GriPhyN would like to stop short of actually

implementing it, but rather act in a consulting role, with LIGO or CACR staff doing the implementation. As part of the collaboration, we will be architecting the security infrastructure, including questions of certificate authority and revocation, interaction with policy committees, delegation of authority. We will evaluate the applicability of the CAS in access control management.

To evaluate the security infrastructure, we will build a parallel, shadow network of gateways into existing LDAS installations. These will be implemented using secure Globus toolkit. Once this network is shown to be working and debugged, we will gradually migrate onto this new fabric and away from our current security solution. (Procedure analogous to how a new freeway is built next to existing highway without disrupting service at any time).

Milestones:

Q1: [UWM] Prototype of basic Grid Security Infrastructure for LDAS: GRAM and GridFTP interface to LDAS. (done)

Q2: [CIT, USC, UWM] Assess LIGO's Virtual Data Grid security needs. Evaluate the applicability of CAS to LIGO's access control needs.

Q3: [CIT, UWM] Implement enhanced security features on existing LDAS installations.

Q4: [CIT, USC, UWM] Demonstration of secure access to multiple LDAS sites.

7 Project Year 2 Milestone Summary By Quarter

Y2 Q1: Oct-Dec 2001:

Demonstration of Virtual Data (GLVD) with a small amount of data in replica catalog, and a small number of channels. (presented at SC'01, November 2001)

Y2 Q2: Jan-Mar 2002:

Building up GLVD prototype to wider quantity of Virtual Data. Implementation of the Transformation Catalog. Assess GLVD security needs, with specific emphasis on CAS.

Y2 Q3: Mar-Jun 2002:

Specification of planning and fault-tolerance requirements. Development of data discovery mechanisms. Implementation of Grid-based security.

Y2 Q4: July-Sept. 2002

Demonstration of Virtual Data technologies developed in Year 2, to be conducted at SC'02. The prototype will be based on the Pulsar search and will include a broad range of Virtual Data products.

SDSS Virtual Data Grid Challenge Problems: Cluster Finding, Correlation Functions and Weak Lensing

James Annis¹, Steve Kent¹, Alex Szalay²

¹Experimental Astrophysics Group, Fermilab

²Department of Physics and Astronomy, The Johns Hopkins University

Draft v1.1 December 17, 2001

We present a roadmap to three SDSS data grid challenge problems. We present some considerations on astronomical virtual data, that the data are relatively complex, that intelligent choices of metadata can speed problem solving considerably, and that bookkeeping is important. The first challenge problem is finding clusters of galaxies in a galaxy catalog, a problem that presents a balanced compute and storage requirement. Second is the computation of correlation functions and power spectra of galaxy distributions, a problem that is compute intensive. Third is the calculation of weak lensing signals from the galaxy images, a problem that is storage intensive. Finally we examine the steps that would lead to grid enabled production of large astronomical surveys.

1	Introduction.....	2
1.1	The SDSS Data Sets	2
1.2	The Challenge Problems	3
1.3	Towards Grid Enabled Production.....	3
2	Astronomical Virtual Data.....	3
2.1	The Data Complexity	3
2.2	Metadata: Design for Speed.....	3
2.3	Derived Data: Design for Variations	4
3	The SDSS Challenge Problems	5
3.1	The SDSS Challenge Problem 1: Cluster Catalog Generation	5
3.2	The SDSS Challenge Problem 2: Spatial Correlation Functions and Power Spectra	6
3.3	The SDSS Challenge Problem 3: Weak Lensing	7
4	The SDSS Experience: Towards Grid Enabled Astronomical Surveys.....	7
4.1	SDSS Data Replication.....	8
4.2	The SDSS Pipeline Experience	8
4.2.1	Description of the Abstracted SDSS Pipeline.....	8
4.2.2	How To Proceed	9
4.2.3	Southern Coadd As A Possible Future Testbed	9
5	Conclusion	11

1 Introduction

The Sloan Digital Sky Survey is a project to map one quarter of the night sky (York et al. 2000). We are using a 150 Megapixel camera to obtain 5-bandpass images of the sky. We then target the 1 million brightest galaxies for spectroscopy, allowing us to produce 3-dimensional maps of the galaxies in the local universe. The final imaging mosaic will be 1 million by 1 million pixels.

This has required the construction of special purpose equipment. Astronomers based at the Apache Point Observatory near White Sands, New Mexico use a specially designed wide field 2.5m telescope to perform both the imaging and the spectroscopy, and a nearby 0.5m telescope to perform the imaging calibration. The camera and spectrographs are the largest in operation. The data reduction operation was designed from the start to handle a very large data set where every pixel is of interest.

We had to learn new techniques to acquire and reduce the data, borrowing from the experience in the high energy physics community. We now need to learn new techniques to analyze the large data sets, and expect to learn together with the GriPhyN collaboration.

1.1 The SDSS Data Sets

The SDSS, during an imaging night, produces data at a 8 Mbytes/s rate. Imaging nights occur perhaps 20-30 nights per year and spectroscopy occupies most of the rest of the nights not dominated by the full moon. Nonetheless, imaging data dominates the data sizes.

The fundamental SDSS data products are shown in table 1. The sizes are for the Northern Survey and the Southern Survey will roughly double the total amount of data.

Table 1: The Data of the SDSS

Data	Description	Data Size (Gigabytes)
Catalogs	Measured parameters of all objects	500
Atlas images	Cutouts about all detected objects	700
Binned sky	Sky after removal of detected objects	350
Masks	Regions of the sky not analyzed	350
Calibration	Calibration information	150
Frames	Complete corrected images	10,000

Our reduction produces complicated data. The catalog entry describing a galaxy has 120 members, including a radial profile. If a different measurement is needed, one can take the atlas images of the object and make a new measurement. If one desires to look for objects undetected by the normal processing, say low surface brightness galaxies, one can examine the binned sky. And one is always free to go back to the reduced image and try a different method of reduction.

The SDSS data sets are representative of the types of data astronomy produces and in particular the types that the NVO will face. We will be working closely with the NVO collaboration.

1.2 *The Challenge Problems*

The SDSS data allow a very wide array of analyses to be performed. Most involve the extraction of small data sets from the total SDSS data set. Some require the whole data, and some of these require computations beyond what is available from a SQL database. We have chosen three analyses to be SDSS challenge problems: these highlight the interesting domain problems of catalog versus pixel analyses, of high computation load, high storage load, and balanced analyses.

1.3 *Towards Grid Enabled Production*

The SDSS data were reduced using custom codes on large laboratory compute clusters. We can learn from the experience what is required to take the reduction into a grid reduction process, something that may be essential to handle the data streams expected from proposed surveys like the Large Synoptic Telescope, which is expected to take data at a rate of Terabytes per night.

2 **Astronomical Virtual Data**

2.1 *The Data Complexity*

Our data is relatively complicated, and we expect that this is a general feature of large surveys. We produce FITS files of the catalog data which contain 120 members including a radial profile (light measurements at a series of aperture sizes) and a large number of arrays (a measurement in each bandpass with an associated error). We produce atlas images, which are the pixel data cut out around detected galaxies, in all five band passes. These are optimal for re-measuring quantities on galaxies. We produce binned sky, which is binned copy of the pixel data for the images with all detected objects removed. The spectroscopy of the objects results in a catalog with another 100 members, and in 1-dimensional images of the spectra that are useful for measuring lines that the production pipeline does not attempt. Finally we have the files in two formats, the flat files in FITS format, and in a database format (two databases, currently).

2.2 *Metadata: Design for Speed*

The metadata requirements for SDSS catalog very naturally map from the concepts of the FastNPoint codes of Andrew Moore and collaborators (Moore et al.). In this world view, Grid Containers are not files or objects, but nodes of a kd-tree (or perhaps some other tree structure with better data insertion properties). In this view what matters for performance is the ability to know what is in the container without having to actually read the contents. Consider a metadata example listing the most useful quantities in astronomy:

Ra, Dec position on sky	bounding box
Z	redshift
r	r-band brightness
g-r	g-r color
r-i	r-i color

If the metadata includes the min, max, mean, standard deviation, and quartiles of the data, the execution time for a range search can be brought down from N^2 to $N \log N$. The central ideas are exclusion (if the range to be searched for does not cross the bounding box, one need not read that container) and subsumption (if the range

to be searched for completely contains the bounding box, one needs the entire catalog, again not reading the container).

Furthermore, there are great possibilities for speed up if one is willing to accept an approximation or a given level of error. Clearly the majority of time in the range search above is spent going through the containers that have bounding boxes crossing the search; it is also true that often this affects the answer but little, as the statistics are dominated by the totally subsumed containers. Having the relevant metadata in principle allows the user to accept a level of error in return for speed.

Often what one is doing is to compare every object against every other object. The tree structure above gives considerable speed up; another comparable speedup is allowed if the objects of interest are themselves in containers with metadata allowing the exclusion and subsumption principles to operate.

These considerations also suggest that a useful derived dataset will be tree structures built against the Grid containers, with the relevant metadata built via time-consuming processing but then available quickly to later users.

2.3 *Derived Data: Design for Variations*

Most of astronomy is derived datasets. One of the clearest examples is the identification of clusters of galaxies. Nature has made a clean break at the scale of galaxies: galaxies and entities smaller are cleanly identifiable as single entities; above galaxies the entities are statistical. Given that, there are many different ways to identify clusters of galaxies. The SDSS currently is exploring 6 different cluster catalogs.

Table 2: SDSS Cluster Finders

Cluster Catalog	Description	Data
MaxBcg	Red luminous galaxies	Imaging catalog
Adaptive Matched Filter	Spatial/luminosity profiles	Imaging catalog
Voronoi	Voronoi tessellation	Imaging catalog
Cut and Enhance	Spatial/color search	Imaging catalog
C4	Color-color space	Imaging catalog
FOG	Velocity space overdensities	Spectroscopic catalog

Each of these catalogs are derived data sets. They may, in principle, be downloaded for existing regions, or the algorithm may be run at individual points in space, or a production run of the algorithm may be scheduled. It is worth pointing out that

- Each algorithms have changeable parameters,
- Each algorithm evolves and hence has version numbers,
- The underlying data can change as the reduction or calibration is re-performed.

We thus point out that versioning and the associated bookkeeping is important.

Finally, we note that generically in astronomy one wishes to attach derived data to the underlying data. Here cluster finding is not a good example, and we will turn to the environment of individual galaxies. View the underlying galaxy data as a table; what astronomers generically wish to do is to add columns. Examples include counting red galaxies in some radius about each galaxy, counting all galaxies in some radius about each galaxy, summing the H-alpha emission from all galaxies with spectra in some radius about each galaxy, etc. The reigning technology in the SDSS is tables with row to row matching by position.

3 The SDSS Challenge Problems

3.1 The SDSS Challenge Problem 1: Cluster Catalog Generation

The identification of clusters of galaxies in the SDSS galaxy catalog is a good example of derived data, and is naturally extendable to the idea of virtual data. We have chosen this as our first challenge problem.

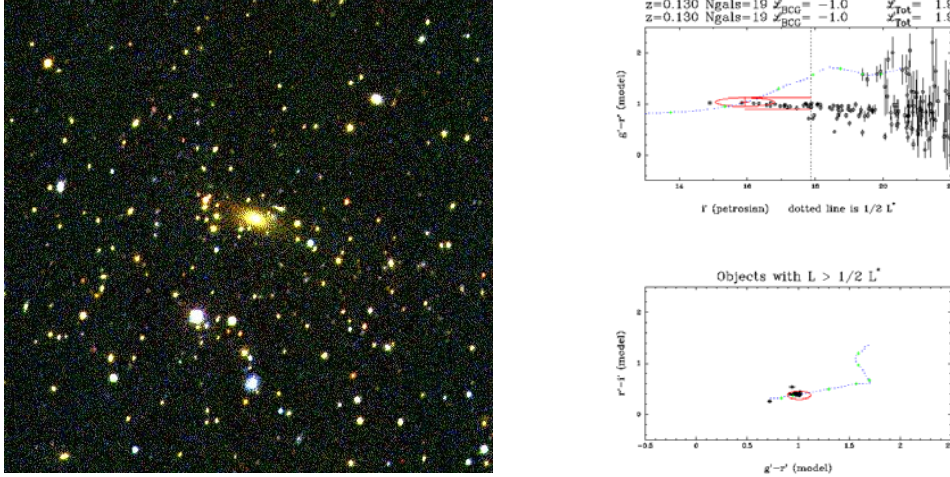


Figure 1: A cluster of galaxies seen in a true color image on the left, and as a color-magnitude and color-color plot on the right. The plots on the right illustrate one cluster finding technique, a matched filter on the E/S0 ridgeline in color-luminosity space: the number of galaxies inside the red is the signal.

Clusters of galaxies are the largest bound structures in the universe; a good analogy is a hot gas cloud, where the molecules are galaxies. By counting clusters at a variety of redshifts as a function of mass, one is able to probe the evolution of structure in the universe. The number of the most massive clusters is a sensitive measure of the mass density Ω_m ; combined with the cosmic microwave background measurements of the shape of the universe, these become a probe of the dark energy.

The basic procedure to find clusters is to count the number of galaxies within some range about a given galaxy. This is an N^2 process, though with the use of metadata stored on trees it can be brought down to a $N \log(N)$ problem. Note that the procedure is done for each galaxy in the catalog.

The problem is computationally expensive, though balanced with the I/O requirements; with the appropriate choices of parameters it can be made either an I/O bound problem or a CPU bound problem. The problem faces moderate storage problems: a hundred square degrees of SDSS data masses to 25 Giga. The problem can be made embarrassingly parallel as there is an outer bound to the apparent size of clusters of interest. The work proceeds through many stages and through many intermediate files that can be used as a form of checkpoint.

The problem is a good choice for the initial challenge problem as

1. cluster catalogs are a good example of derived data,

2. cluster catalog creation is roughly compute and storage balanced, and
3. it can be solved in interesting times on existing testbeds.

In terms of using GriPhyN tools, it exercises

1. the use of derived data catalogs and metadata,
2. replica catalogs,
3. transformation catalogs including DAG creation, and
4. to use existing cluster finding code advances in code migration must be made.

3.2 *The SDSS Challenge Problem 2: Spatial Correlation Functions and Power Spectra*

Our second challenge problem is aimed at computationally expensive measurements on catalog level data. We choose measurements on the spatial distribution of galaxies, which contain interesting cosmological information.

The correlation function of the positions of galaxies projected on the sky forms a Fourier transform pair with the spatial power spectrum. The power spectrum itself is of great interest in so much as the light from stars in galaxies traces the underlying mass both of normal matter and of the dark matter. If light traces mass, then when one measures the power spectra of galaxies one is measuring the power spectra of mass in the universe. The power spectrum may be predicted theoretically given a cosmology and mass and energy content of the universe, and thus this measurement explores very interesting quantities. The main uncertainty here is that it is known that the distribution of galaxies is biased away from the distribution of mass; exactly how much is a matter of some debate. The SDSS will allow these correlation functions to be measured and analyzed as a function of galaxy properties (e.g. magnitude, surface brightness, spectral type).

If the redshift of the objects are known, either from spectroscopy or by photometric redshift techniques, one is able to compute the power spectrum directly. This often involves an expensive SVD matrix calculation. The same push to measure the power spectrum as a function of galaxy properties exists, for the same reason; the relation of the galaxies to the underlying mass is uncertain.

The essential procedure is to count the distance from each galaxy to every other galaxy, accumulating the statistics. This is an N^2 process (and higher order correlations equivalently higher order) though again metadata employing tree structures can cut the expense down to $N \log(N)$. The SVD matrix calculation is of order N^3 . Neither program is particularly parallelizable, only made embarrassingly parallel by placing an arbitrary large angle cutoff. For the correlation function there is a further expensive step, that of the extensive randomization procedure that must be carried out in order to establish the statistical significance of the measurements.

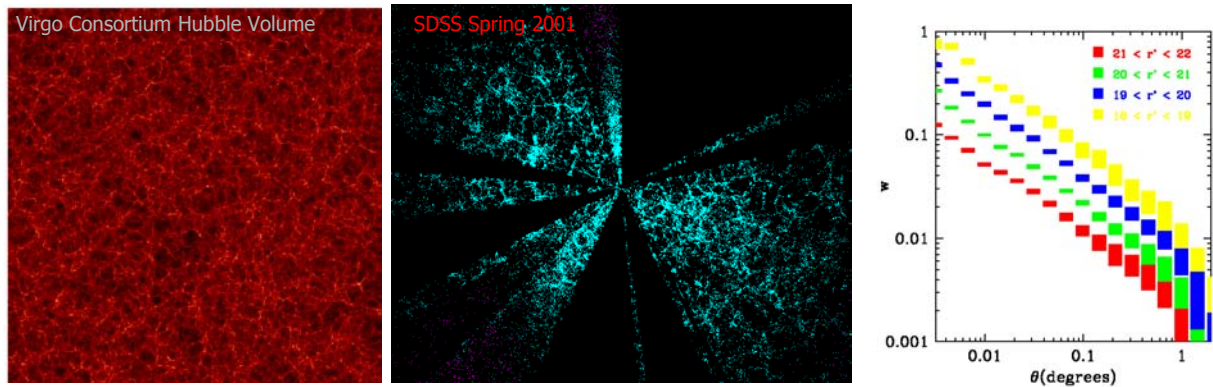


Figure 2: The correlation function. On the left panel, a numerical simulation of a Gigaparsec scale. In the middle, the SDSS redshift survey over a similar scale. On the right, the correlation function from the angular distribution of galaxies. The distribution of galaxies on the sky and in redshift contains cosmological information.

In both the correlation function measurement and the power spectrum measurement, the computation burden dominates the storage burden.

The correlation function challenge problem is an example of a catalog level analysis requiring large amounts of compute time. Each correlation function is computationally expensive, and very often the results are used as input to another layer of sophisticated codes. Correlation functions are an example of virtual data where a premium will be placed on locating existing materializations before requesting an expensive new materialization. If none can be found, then one enters the realm of resource discovery and job management. Locating existing materializations that are relevant is of course a very interesting problem: how does one construct a GriPhyN Google?

3.3 The SDSS Challenge Problem 3: Weak Lensing

Our third challenge problem is aimed at the storage bound problem of making new measurements on the pixel level data. We choose the problem of making optimal moments measurements of galaxies in the atlas images, which can be used to make weak lensing measurements of mass at cosmological distances.

Weak lensing is caused by mass distorting the path of light from background objects; the objects tend to align in concentric circles. Tend is the operative phrase; weak lensing is an inherently statistical program. One use for weak lensing is to measure the masses of the clusters. In the North one averages many clusters together to get signal. In the SDSS South, which is made considerably deeper by coadding many images together, individual clusters may have their mass measured. Two other measurements are equally interesting: measurements of the mass of the average galaxy, binned by interesting quantities, and measurements of the power spectrum of large scale structure induced weak lensing fluctuations.

In order to do weak lensing, one must make suitably weighted second moments analysis of each image. Most of the algorithmic magic lies in the exact weighting, some in the details of the moment analysis. This is an N^2 process in the number of pixels, which of course is much larger than the number of galaxies. Despite this, the problem is weighted towards storage. The vast bulk of the atlas images that must be shepherded about dominates the problem. An analysis of the problem must include whether bandwidth is limited; if it is, compute power local to the data is preferred, if not, then the access to the distributed computing would be preferred.

The weak lensing challenge problem is a pixel level analysis that requires the moving of large amounts of data. Work will need to be done on data discovery, data and resource management.

4 The SDSS Experience: Towards Grid Enabled Astronomical Surveys

In the long term, it is of interest to use GriPhyN tools in full scale production. The following two sections describe issues in production that map naturally onto Grid technologies.

4.1 *SDSS Data Replication*

Several sites wish to have local copies of the SDSS. Mostly this wish extends only to the catalog data, which will mass about a Terabyte at the end of the survey. We can expect that soon astronomers will wish to mirror the atlas images or even the reduced images.

The current model for transferring the catalogs involves sending tapes and sending disks. This is not without problems. The data live in the Objectivity based database SX, and the model has been to send loading files that the remote sites can load into their own SX database.

A different model would be to employ GDMP, which is nearly ideal for the purpose. It implements a subscription model that is well matched to the problem. It uses an efficient FTP, important when faced with 100 Gig scale transfers. It has transfer restart capability, very important when faced with the small pipe to Japan. The main problem here will be to convince our colleagues to spend the energy to bring up Globus and GDMP. Unless it is very easy to install and run, astronomers will return to the existing, if non-optimal, tools.

4.2 *The SDSS Pipeline Experience*

Data processing in SDSS is procedure oriented: start with raw data, run multi-stage processing programs, and save output files.

The data processing is file oriented. While we use the object-oriented database SX, the objects are used only internally. The natural unit of data for most SDSS astronomers is the “field”, one image of the sky, whose corresponding catalog has roughly 500 objects in it.

4.2.1 *Description of the Abstracted SDSS Pipeline*

The SDSS factory itself lives in the DP scripts that join the pipelines. There are 3 generic stages of the factory at each and every pipeline:

INPUTS:

1. A plan file - defining which set of data are to be processed.
2. Parameter files - tuning parameters applicable to this particular set of data.
3. Input files - that are the products of upstream pipelines.
4. Environment variables - defining which versions of pipelines are being run.

COMPUTATION:

1. Prep
 - a. generate plan file containing, for example, root directories for input and output
 - b. locate space
 - c. make relevant directories
 - d. stage the input data
 - e. make relevant sym links
 - f. register the resources reserved in a flat file database
 - g. Call submit
2. Submit
 - a. generate a shell script the fires off the batch system submit
 - b. Call ender
3. Ender

- a. Run a status verify job that checks if the submitted job did complete. The existence of the files that should have been created is necessary and almost sufficient for the next step to succeed.
- b. Run a pipeline verify job to generate QC information (e.g., the number of galaxies/image) that are given a sanity check. If success, call Prep for the follow-on pipeline.
- c. Run a "scrub" job that removes most files once they have been archived. The archiving is itself a "pipeline".

OUTPUTS:

1. Output files - the data products themselves.
2. Log and error files - log and error files
3. Quality control files - identify outputs so fatally flawed that subsequent pipelines cannot run.
4. A single status flag – 0, proceed to next pipeline, 1, hand intervention required.

These are daisy chained: the first invocation of Prep takes as an argument how many of the following pipelines to run in series.

4.2.2 How To Proceed

We note that the abstracted SDSS pipeline maps very well onto the DAG concepts of Condor. The path to the future lies in understanding the value added of using the DAG technology.

4.2.3 Southern Coadd As A Possible Future Testbed

The SDSS Southern Survey will in the end coadd various imaging runs on the same piece of sky to produce deep images. These images are suitable for making weak lensing measurements on individual clusters, as well as making more detailed weak lensing measurements on galaxies and large scale structure.

If one chose the problem of building weak lensing maps, one would exercise all of the pipeline structure outlined above. One vision of the SDSS/NVO team is to spin the SDSS data on a compute cluster and allow for demand driven re-reduction with the ever-improving versions of Photo. One cannot just run Photo, as it is the center of a long processing chain, but it is exactly this processing chain that must be made demand driven to solve the weak lensing map problem.

4.2.3.1 Data Sizes

The components of any given 200 sq-degree coadd of the south come to:

catalogs	measured parameters of all objects	10 Gig
atlas images	cutouts around all objects	15 Gig
binned sky	sky leftover after cutouts, binned 4x4	7 Gig
masks	regions of the sky not analyzed	7 Gig
calibration	a variety of calibration information	3 Gig
frames	corrected images	Nx200 Gig

4.2.3.2 Design Detail

1. given an area on which to coadd,
 - a. Take user input
2. find relevant reduced data
 - a. Query either a db or a flat file for the the runs that have been observed and extract the list of runs that are of the right strip.
 - b. Using that list, determine which runs overlap the given area. Involves the calculation of a spatial intersection.
 - c. Apply cuts against data that are not of high enough quality to use (even though other parts of the run may be.)
 - d. Estimate how much data is involved.

Metadata: a list of runs and portions of runs involved
3. find disk space and compute power for input data, processing, and outputs
 - a. Query the local network for available machines
 - b. Query the local network for available storage
 - c. Reserve the resources for a period of time

Metadata: a list of machines
4. extract relevant reduced data from storage
 - a. From some knowledge base, determine for each run if the data is:
 - 1) on archival disk
 - 2) on production disk
 - 3) on fast tape
 - 4) on slow tape
 - b. Arrange to get the data off the media and onto the production machines

Metadata: a list of portions of runs and where they live on production disk
5. build mapping function from calibration data
 - a. An image is a 1361x2048 array of pixels. In general two images of the "same" piece of sky will be:
 - 1) slightly offset in both x and y
 - 2) slightly rotated
 - 3) have different small distortions as a function of x,y superimposed
 - 4) have different conversion between pixel value and energy flux
 - b. These translations, rotations, distortions, and scalings are calculateable from a set of calibration information that may or may not be kept as a header to the data itself.
 - c. Find the calibrations, and build the mapping function for each pixel

Metadata: computed mapping functions to be applied to the data
6. perform coadd to create new reduced data
 - a. Load a set of co-located images into memory
 - b. Apply mapping function
 - c. Perform median or average on stack of pixels (in truth: apply very clever algorithm to make maximal use of information and minimize noise)

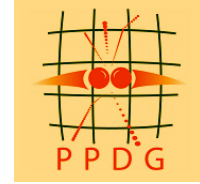
- d. Save output to disk.
Metadata: Location of output data
- 7. run full SDSS pipeline on new data set
 - a. arrange for all necessary input files to be in place
 - b. arrange for all 10 pipelines to run
 - c. watch success or failure of the pipelines
 - d. save resulting outputs to disk
 Metadata: Location of output data
- 8. keep track of intermediate coadd catalogs and data
 - a. The output of 6. is to be preserved, as 7. can be done multiple times
 - b. The output of 7. is to be preserved
 - c. Each time new data comes in, the above is done.
- 9) Run weak lensing analysis on the intermediate coadded south
 - a) locate atlas images on disk
 - b) compute optimal shape parameter
 - c) produce shape catalog

5 Conclusion

We have outlined the data of the SDSS, and how we might approach it with GriPhyN tools. Starting with cluster finding as an initial virtual data problem, we move through correlation function and power spectrum problems which push computational and virtual data recovery issues, to a weak lensing analysis on the pixels which push data management issues. We then lay out possible avenues for work leading to the incorporation of Grid tools in survey production.



Technical Report GriPhyN-2001-15
www.griphyn.org



DRAFT: COMMENTS SOLICITED

GriPhyN/PPDG Data Grid Architecture, Toolkit, and Roadmap — Version 2 —

The GriPhyN and PPDG Collaborations

Editors: Ian Foster, Carl Kesselman
(foster@mcs.anl.gov, carl@isi.edu)

Contributors: Ewa Deelman, Ian Foster, Carl Kesselman, Harvey Newman, Doug Olson, Ruth Pordes, Richard Mount, Alain Roy, Mike Wilde

Abstract

Data Grids are emerging as important to a range of scientific and engineering disciplines that depend for their progress on community creation, curation, and often computationally intensive analysis of large quantities of data. GriPhyN and PPDG are two major U.S. research and development projects aimed at creating and realizing the promise of Data Grids in practical settings, emphasizing in particular the requirements of physics and astronomy applications but also working closely with other domains. This document provides a comprehensive statement of our current understanding of the *requirements* that motivate development of Data Grids and then addresses three aspects of our approach to addressing these requirements: (1) the *architecture* that has been defined by the GriPhyN and PPDG projects to meet these requirements; (2) the current state of our instantiation of this architecture as realized in the GriPhyN *Virtual Data Toolkit* and PPDG experiment end-to-end applications, and (3) a *roadmap* that defines both the path that we expect the projects to follow to extend the reach of the toolkit and application components, and areas in which no work is currently planned (and that hence represent areas where new projects are perhaps needed). The document is intended to be a complete treatment of these issues, although in many cases it refers to other documents for technical details concerning APIs, protocols, and requirements. This is the second version of this document, which will evolve through a series of revisions for the foreseeable future.

Table of Contents

1	Introduction.....	3
2	Requirements	4
2.1	General Statement of Requirements	4
2.2	Numerical Requirements and Challenges	5
3	Data Grid Architecture Elements.....	6
3.1	Workflow View	6
3.2	Architecture Overview.....	9
4	Basic Grid Protocols	10
4.1	Communication.....	10
4.2	Authentication and Authorization.....	11
4.3	Resource Discovery and Monitoring	11
4.4	Resource Management.....	12
5	Data Grid Resources	13
5.1	Storage Systems	13
5.2	Compute System	15
5.3	Network.....	16
5.4	Code Repositories	16
5.5	Catalog Services.....	17
6	Request Planning and Execution Services.....	19
6.1	Abstract and Concrete Representations	20
6.2	Grid Directed Acyclic Graphs	20
6.3	Request Planning Services.....	23
6.4	Request Execution Services.....	23
7	Information Services	24
7.1	Discovery and Monitoring Infrastructure	25
7.2	Collective Monitoring Services	25
7.3	Replica Location Service	25
7.4	Catalog Management Services.....	25
7.5	Grid Container Management Services(GCMS)	26
8	Policy and Security Services.....	26
8.1	Community Authorization Service	27
8.2	PKI Scalability and Usability.....	27
9	Replication	27
9.1	Replica Management Services.....	27
9.2	Code Distribution.....	29
10	Performance Estimation and Evaluation.....	29
11	Planning, Execution and Error Recovery.....	29
12	Missing Components	29
	Acknowledgments.....	29
	Bibliography	29

1 Introduction

The term “Grid” refers to technologies and infrastructure that enable *coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations* [12, 14]. This sharing relates primarily to direct access to computers, software, data, networks, storage and other resources, as is required by a range of collaborative computational problem-solving and resource-brokering strategies emerging in industry, science, and engineering.

Grid concepts are particularly relevant to data-intensive science due to the collaborative nature of data production and analysis, and the increasing complexity of data analysis tasks, which, increasingly, requires the harnessing of large distributed collections of shared resources. We thus see considerable interest in the creation of so-called *Data Grids*, i.e., Grid infrastructures, tools, and applications designed to enable distributed access to, and analysis of, large amounts of data.

Within the U.S., the NSF-funded Grid Physics Network (GriPhyN, www.griphyn.org) and the DOE-funded Earth System Grid (ESG, www.earthsystemgrid.org) and Particle Physics Data Grid (PPDG, www.ppdg.net) projects are working together to develop a comprehensive set of Data Grid technologies to be deployed on a large scale by a set of partner science projects. (In Europe, the EU DataGrid, www.eu-datagrid.org, project is pursuing similar goals.) The partner science projects targeted by GriPhyN, ESG, and PPDG are, initially, experiments in high energy and nuclear physics, astronomy, and climate science, although there are also close ties with other science and engineering disciplines.

GriPhyN and PPDG partners are engaged *now* in developing next-generation Grid-based data analysis systems. It is hence important to document clearly the requirements that the GriPhyN/PPDG software is intended to address, the overall architecture of this software, and the APIs and protocols to which the software adheres. It is also important to document planned enhancements to the software and to identify major functionality that is missing—and that hence represents opportunities for contributions by other partners, and/or future collaborative development.

This document is intended to provide in one place a comprehensive review of requirements, architecture, current state, and future plans for the GriPhyN and PPDG common software suite. In many cases, this review refers to other documents for technical details, but we aim to be complete in terms of documenting current understanding of the issues that must be addressed in a Data Grid system, and the current status of our “production” software. Note that in addition to the common software documented here, there will always be other software components that are being used by individual experiments or projects. These may eventually appear into subsequent versions of this document, if they generate widespread interest.

The Data Grid architecture has a highly modular structure, defining a variety of components, each with its own protocol and/or application programmer interfaces (APIs). These various components are designed for use in an integrated fashion, as part of a comprehensive Data Grid architecture. However, the modular structure facilitates integration with discipline-specific systems that already have substantial investments in data management systems.

This second version of this document represents a significant evolution over the first version [11]. Further versions will be produced from time to time as our software, and understanding, evolves.

2 Requirements

2.1 General Statement of Requirements

We provide here some background information, taken in part from [7], on the problems we wish to solve. The experiments targeted by GriPhyN and PPDG (ATLAS, BaBar, CMS, D0, JLAB, LIGO, SDSS, STAR, TJNF, and NVO [29]) share with other data-intensive applications a need to manage and process large amounts of data [6, 21]. This data comprises raw (measured) and many levels of processed or refined data as well as comprehensive metadata describing, for example, how the data was generated, how large it is, etc. The total size of this data ranges from terabytes (in the case of SDSS) to petabytes (in the case of ATLAS and CMS [S. Bethke (Chair), 2001 #1757]).

The computational and data management problems encountered in these experiments include the following challenging aspects:

- *Computation-intensive as well as data-intensive:* Analysis tasks are compute-intensive and data-intensive and can involve thousands of computer, data handling, and network resources. The central problem is coordinated management of computation and data, not just data curation and movement.
- *Need for large-scale coordination without centralized control:* Stringent performance goals require coordinated management of numerous resources, yet these resources are, for both technical and strategic reasons, highly distributed and not amenable to tight centralized control.
- *Large dynamic range in user demands and resource capabilities:* We must be able to support and arbitrate among a complex task mix of experiment-wide, group-oriented, and (thousands of) individual activities—using I/O channels, local area networks, and wide area networks that span several distance scales.
- *Data and resource sharing:* Large dynamic communities would like to benefit from the advantages of intra and inter community sharing of data products and the resources needed to produce and store them.

We introduce the *Data Grid* as a unifying concept to describe the new technologies required to support such next-generation data-intensive applications—technologies that will be critical to future data-intensive computing not only in the physics experiments mentioned above, but in the many areas of science and commerce in which sophisticated software must harness large amounts of computing, communication and storage resources to extract information from measured data. We use the term Data Grid to capture the following unique characteristics:

- A Data Grid has *large extent*—national or worldwide—and *scale*, incorporating large numbers of resources and users on multiple distance scales.
- A Data Grid is more than a network: it layers sophisticated *new services* on top of local policies, mechanisms, and interfaces, so that geographically remote resources (hardware, software and data) can be shared in a coordinated fashion.
- A Data Grid provides a new degree of *transparency* in how data-handling and processing capabilities are integrated to deliver data products to end-user applications, so that requests for such products are easily mapped into computation and/or data retrieval at multiple locations. (This transparency is needed to enable sharing and optimization across diverse, distributed resources, and to keep application development manageable.)



As we shall see, transparency is an important aspect of our Data Grid architecture. In its most general form, transparent access can enable the definition and delivery of a potentially unlimited virtual space of data products derived from other data. In this virtual space, requests can be satisfied via direct retrieval of materialized products and/or computation, with local and global resource management, policy, and security constraints determining the strategy used. The concept of *virtual data* recognizes that all except irreproducible raw measured data (or computed data that cannot easily be recomputed) need to ‘exist’ physically only as the specification for how they may be derived. The grid may materialize zero, one, or many replicas of derivable data depending on probable demand and the relative costs of computation, storage, and transport. In high-energy physics today, we estimate that over 90% of data access is to derived data.

We note that virtual data as defined here integrates familiar concepts of data replication with the new concept of data generated “on-the-fly” (“derived”) in response to user requests. The latter is a research problem while the former is common practice. Furthermore, the two concepts are logically distinct: it can be useful and important to replicate data even if derived data is not being generated automatically. However, we believe that there are likely to be advantages to an integrated treatment of the two concepts. For example, an integrated treatment will make it possible for users to choose to regenerate derived data locally even if the data is already materialized at a remote location, if data regeneration is faster than data movement.

2.2 Numerical Requirements and Challenges

The preceding discussion emphasizes that Data Grid systems must be able to support a wide range of environments and applications. Nevertheless, it is useful to characterize in approximate terms the scale of the problems to be addressed, in order to identify areas in which we face the most significant challenges. Table 1, taken from the GriPhyN proposal, represents one attempt to summarize these requirements.

Table 1: Characteristics of the physics experiments targeted by GriPhyN and PPDG

Appli- cation	First Data	Data Rate MB/s	Total? Raw? Data Volume (TB/yr)	User Comm- unity	Data Access Pattern	Compute Rate (Gflops)	Type of data
SDSS	1999	8	10	100s	Object access and streaming	1 to 50	Catalogs, image files
LIGO	2002	10	250	100s	Random, 100 MB streaming	50 to 10,000	Multiple channel time series, Fourier transformations
BaBar	2000	15	500	500	Object Database	TBD	Raw and processed events (?/event) need official numbers
D0	2001	15	350	500	Sequential access to files	TBD	Raw and processed events (250Kbytes/event)
STAR	2000	TBD	TBD	TBD	TBD	TBD	TBD
JLAB	2000	TBD	TBD	TBD	TBD	TBD	TBD
ATLAS/ CMS	2006	100	5000	1000s	Streaming, 1 MB object access	120,000	Events, 100 GB/sec simultaneous access

The experiments taking data today—SDSS, STAR, BaBar and D0—have existing data analysis systems that must be extended to provide the distributed computation and data transport required in the above table. The Data Grid services being developed must be integrated carefully into these systems to maintain their continuous operability and performance.

The following are more detailed estimates of the scope and performance demands associated with an LHC experiment around 2005:

- 200 sites with significant computing and storage resources (“central” and “regional” centers).
- A total of 30,000 computers and 15 PB of storage.
- 5 PB of unique data.
- 50 million logical files
- 500 million physical files
- 20 million physical files catalogued at a (large) site
- Average query time for the replica location service of 10 milliseconds
- Maximum query time for the replica location service of less than 5 seconds
- Replica location service query rates of 10 to 100 per second (burst)
- Replica location service query update/insertion rate of 5 to 20 per second (burst)

We can also point to application scenarios that could lead to significant increases in requirements in several areas. [Need to point to the monarc documents here](http://home.cern.ch/~barone/RC/RCA5.txt)

<http://home.cern.ch/~barone/RC/RCA5.txt>

http://mcs.web.cern.ch/MONARC/docs/monarc_docs/1999-02.html For example, the resource estimates for LHC experiments assume that we are only harnessing resources at a modest number of dedicated regional computing centers. Now it could be that with appropriate software, we could increase accessible computing power dramatically by harnessing idle workstations at all participating institutions. Changes in the event trigger rate can also cause significant changes in the resource requirements.

3 Data Grid Architecture Elements

The discussion above indicates some of the principal elements that we can expect to see in a Data Grid architecture. In this section, we start the process of translating this general discussion into specifics, first by presenting a workflow view of Data Grid operations and then by an analysis of the different types of services required in a system that supports this workflow.

3.1 Workflow View

Figure 1 presents one view of the major elements of a typical Data Grid end-to-end application. In this picture, the elements inside the large lower box are intended to be domain-independent, Data Grid services.

1. An *application* workflow specification invoked by a Data Grid user specifies a set of Data Grid operations that are to be performed. In the proposed architecture, application workflows are specified via an abstract Grid Directed Acyclic Graph (G-DAG) which provides a standard syntax and semantics for representing Grid operations. The G-DAG

representation is discussed in Section 6.2. This request will, typically, be somewhat abstract, referring to data products without regard to their location and/or materialization.

2. The abstract G-DAG is passed to a *request planner* that translates the original request into a *concrete* G-DAG representing a more detailed plan specifying data movement operations and computations. In the course of developing this G-DAG, the request planner may consult catalogs, information services, and so forth. We note that the degree to which this G-DAG specifies concrete actions vs. leaving decisions to execution is a topic of debate; our intention is to define a framework in which various alternative strategies can be investigated. The request planner needs to interface with the user/application for example to provide information about the estimated duration of the plan execution. It is possible, that the estimated execution time of the amount of resources needed to execute the plan are too large for a given user/application.
3. The G-DAG is passed to a *request executor* that manages the execution of the request on the Data Grid, potentially locating resources, mapping operations to resources, tracking progress of operations, handling various failure conditions, and so forth. As computation proceeds, the request executor may also update various catalogs, for example to record that additional data products have been materialized and/or replicas produced, and to document the procedures followed to produce new data products.
4. The request executor may call upon other services, for example a *reliable data transfer service* to orchestrate data movement and/or create additional data replicas and *compute services* to perform computation.

In addition, various system components operate:

- *Accounting* systems keep track of resource usage, making it available for purposes of reporting, tracking, policy enforcement, and so forth.
- *Monitoring* systems perform active monitoring for purposes, for example, of intrusion and anomaly detection.
- *Replica creation* systems monitor system behavior with the goal of identifying when and where replicas should be created (or deleted), and generate job requests appropriately.

It is possible that the mechanisms in place for the request executor will be insufficient to deal with failures, for example if the derived data assumed to exist is not found. In such cases, the request executor needs to interact with the request planner to generate an alternate execution scenario.

In support of these components, we have a variety of services and resources:

- *Data catalogs* maintain information about data itself (metadata), the transformations required to generate derived data (if employed), and the physical location of that data.
- *Community authorization services* maintain information about the policies that govern who can use what resources for what purposes, and enforce these policies.
- The various *resources* that we must deal with: storage systems, computers, networks, catalogs, and code repositories. (In the rest of this text, we use the word “resource” to refer to this range of entities, unless otherwise noted.). These resources may include generic Grid resources such as compute and storage servers, or application specific resources and services such as database servers.
- An *information service* supports the discovery and monitoring of Grid resources as well as application specific services and resources.

Notice the clean separation of concerns expressed in Figure 1 between virtual data (catalogs) and the control elements that operate on that data (request manager). This separation represents an important architectural design decision that adds significantly to the flexibility of a Data Grid system.

In addition, we of course have the diverse and complex software systems constructed by partner science projects to perform their data collection and analysis. While this software is represented in Figure 1 as a small box, in most applications (e.g., those found in high energy physics) considerable development may well be required at this level in order to construct application systems that can exploit Data Grid mechanisms effectively. For example, we can have sophisticated application-specific *request formulation tools* that enable the end user to define data requests, translating from domain-specific forms to standard request formats, perhaps consulting application-specific ontologies. However, this machinery is beyond the scope of this document. (Unless someone wants to volunteer to develop requirements and specifications.)

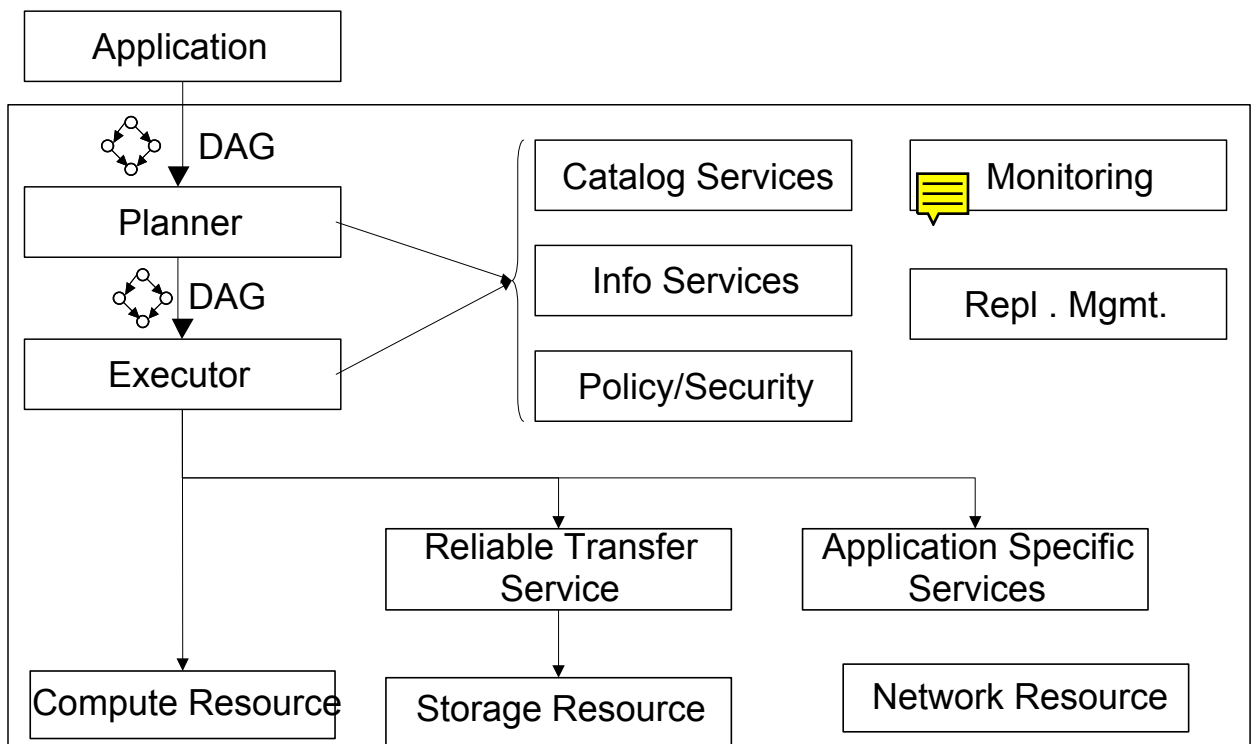


Figure 1: Schematic showing the principal elements of the Data Grid reference architecture ITF: What else do we need if we want to be comprehensive? E.g., we don't show code repositories. .

The architecture depicted in Figure 1 provides the flexibility needed to enable data management as envisioned by the experiments. For example, in CMS, a job is described as a set of possibly many subjobs (each being one executable)

(<http://kholtman.home.cern.ch/kholtman/griphynaug23.ppt> [there might be a better reference, such as a doc]). The subjob descriptions are location independent. The data flow between subjobs is represented in terms of file sets, which are specified in terms of logical file names. Error rules are also provided in the subjob descriptions. The grid scheduler optimizes job execution by mapping subjobs to sites, generating any necessary data replication instructions.



Grid-wide execution service can use error recovery rules to recover automatically from common classes of failure.

3.2 Architecture Overview

Experience with a range of Grid applications and architectures tells us that the application-specific view of Data Grid architecture does not tell the whole story. When accessing storage or compute resources, we require resource discovery and security mechanisms. When developing request plans, we need to be able to capture and manipulate such plans. We also require logic for moving data reliably from place to place, for scheduling sets of computational and data movement operations, and for monitoring the entire system for faults and for responding to those faults. These various mechanisms and services are for the most part application-independent and can be developed as independent mechanisms and reused in different contexts.

These observations lead us to view Data Grid mechanisms as being not independent of, but rather built on top of, a general Grid infrastructure [14]. Thus, a Data Grid system is defined, first of all, by a set of *basic Grid protocols* used for data movement, name resolution, authentication, authorization, resource discovery, resource management, and the like. These basic protocols underpin all other Grid services and, as discussed in [14], are the basis for interoperability between Data Grid systems and with other Grid deployments. We adopt the protocols defined by the Globus Toolkit as they represent the de facto standard for Grid systems. In terms of Figure 2 (from [14]) these protocols comprise the Connectivity and Resource layer.

Building on this base, a Data Grid provides the programmer with three general classes of component:

- We have the *resource services* that provide Data Grid applications with access to individual Grid resources, whether computing, storage, network, or other physical devices. These services are defined by the protocols used to invoke them and by the behaviors that they exhibit in response to protocol messages. As we shall explain, Data Grids can exploit standard protocols but can benefit from specialized behaviors in the Fabric elements to which these services provide access. (These are the services provided by the Resource layer in Figure 2.)
- We have the higher-level *collective services* that support the management and coordinated use of multiple resources. Here, Data Grids introduce specialized requirements, in such areas as catalog services, replica management services, community policy services, coherency control mechanisms for replicated data, request formulation and management functions for defining, planning and executing virtual data requests, and replica selection mechanisms. We can also reuse standard collective services, for example, for resource discovery. (This is the Collective layer of Figure 2.)
- Finally, we have the various *programming tools* used to construct applications, and of course the applications themselves.

We structure the discussion that follows in terms of these categories.

Specific applications exploit Data Grid capabilities by invoking services at the Collective or Resource level, typically via supplied APIs and SDKs that speak the protocols used to access

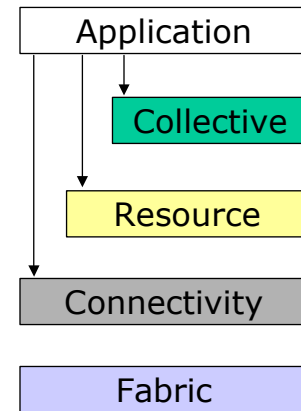


Figure 2: Schematic of Grid architecture

those services. To be successful, our architecture must make it feasible to adapt this experiment-specific software for Grid operation not by rewriting it but rather by layering it on top of the services and APIs provided by the elements just described (request manager, catalogs, Grid services, etc.). Reconciling the demands of the experiments with the capabilities provided by Data Grid software is an ongoing process of negotiation and experiment.

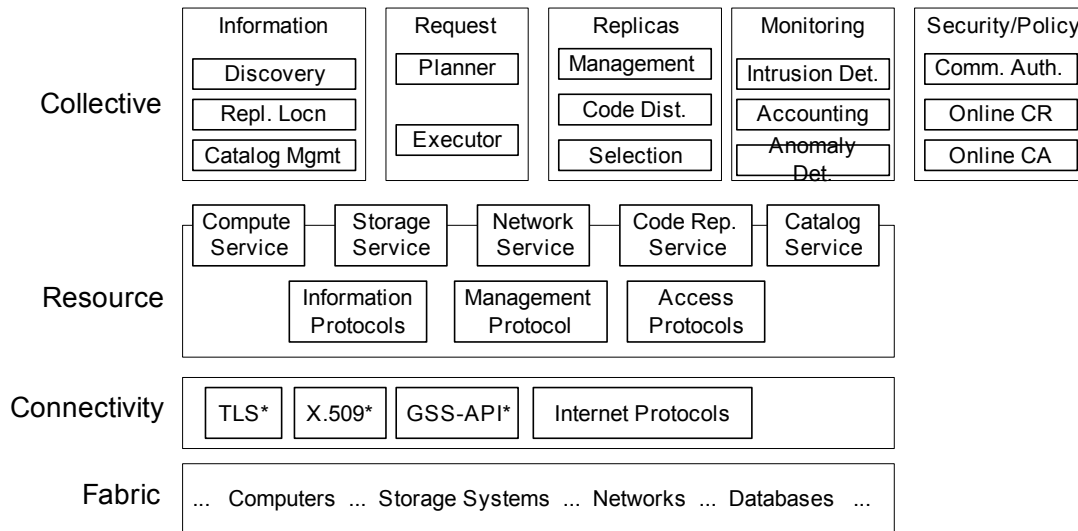


Figure 3: A far-from-complete listing of major Data Grid Reference Architecture elements, showing how they relate to other Grid services. Shading indicates some of the elements that must be developed specifically to support Data Grids.

4 Basic Grid Protocols

We assume the existence of general-purpose protocols for communication, authentication, resource discovery, and resource management. We describe these protocols briefly here as background for the discussion that follows. While these elements are invisible to the Data Grid user, they are important from the perspective of Data Grid developers, as they define the “neck” in the protocol hourglass, below which can be placed many different resources and above which can be implemented many services. For example, standard authentication mechanisms allow different Data Grid developers to develop different components secure in the knowledge that there are standard mechanisms for establishing the identity of users and resources. Similarly, a standard access protocol for storage elements allows developers of storage systems and developers of higher-level capabilities (e.g., replica management) to work independently.

4.1 Communication

Data Grid applications require communication mechanisms that can, depending on context, provide for secure, high-speed, and/or performance-guaranteed transmission. The basic Internet protocols have deficiencies in each of these areas, but we can nevertheless work with them while waiting for improved capabilities. Hence, our strategy in this area is to use basic Internet protocols for communication, name resolution, and the like. As advanced capabilities such as IPv6, IPsec, and quality of service (QoS) become generally available, we should investigate their utility for Data Grid applications.

Current status: Considerable experimentation with QoS has been conducted (e.g., [15]), but lack of broad deployment has so far prohibited any practical application.

Future plans: Integrate IPv6 support into Grid communication libraries. Experiment with wavelength allocation in WDM networks.

4.2 Authentication and Authorization

Data Grid requirements for authentication and authorization include:

- Secure, bilateral authentication of users and resources: i.e., mechanisms that allow two entities (e.g., user+service, user+resource, service+resource) to verify each other's identity.
- Confidentiality of data transferred.
- Access control (authorization) expressible by resource, code, and data owners. This corresponds to enforcement of local policy. Examples of such local policy may be file-system and CPU usage quotas, usage priorities, ACLs, etc.

Other requirements relating to community authorization and the publication and discovery of access control policies are discussed later.

We address these requirements by adopting the public key infrastructure (PKI)-based Grid Security Infrastructure (GSI) [13], which provides the following essential capabilities:

- *Single sign on*, allowing a user to authenticate once and then delegate rights to a computation that runs on their behalf, accessing resources without further requiring the user to engage in further authentication operations.
- *Mutual authentication* of computations and resources.
- *Authorization* via call outs to local policy.
- *Message privacy and integrity*, if required.

These services are accessible via a standard API (GSS-API), from the GlobusIO package for secure socket-based communication, or from the many tools that have integrated GSI support. Other supporting tools address credential management.

Current status: GSI is broadly adopted and GSI-enabled versions of a wide variety of tools exist, including FTP servers and clients, Secure Shell, Condor [19], SRB, etc. Discussions of standard Certificate Authority (CA) policies are ongoing in GGF.

Technical details: See <http://www.globus.org/security/> for details on GSI.

Future plans: An R&D project funded by DOE's SciDAC program is continuing to evolve GSI, addressing issues of usability and scalability (credential management), restricted delegation, etc. Reach agreement across Data Grid and other Grid projects (e.g., DTF) on standard CA policies. Address issues of community policy, as discussed in Section 8.1 below.

4.3 Resource Discovery and Monitoring

We next require mechanisms that support the discovery, characterization, and monitoring of Grid resources. We adopt those defined and implemented by the Globus Toolkit's Meta Directory Service (MDS-2).

- A registration protocol, the Grid Resource Registration Protocol (GRRP), that allows a resource or service (yet more generically, a "sensor") to notify another entity of its availability and how to contact it for purposes of enquiry or control. Such a protocol

might be used, for example, to register resources with an information service that then constructs an index of known storage systems.

- An inquiry protocol, the Grid Resource Inquiry Protocol (GRIP), used to enquire as to the structure and state of a resource or service. A data model and query language are also defined. For example, in the case of a storage system, an enquiry protocol might allow an application to query a storage system concerning its capacity, speed, availability, and functionality.

In brief, we require that all resources and services in a Data Grid speak these protocols, hence making it possible for us to discover them and determine their properties. More details on this technology can be found in [8] (and in [17] for GRRP) and at www.globus.org/mds.

Given the basic protocols a wide variety of different information organization and distribution schemes can be created via the construction of information indexes. Indexes may be queried via Grid Information Protocols (GRIP) or make support specialized query protocols such as ODBC.

Current status: MDS-2.1 defines data models and provides implementations of sensors for computers, storage systems and several other devices. An interface to the Network Weather Service (NWS) [31] provides network performance information. Recently, sensors have been developed for GridFTP servers, that provide information on transfer performance. GSI security is supported for accessing MDS-2.1 components and information provided by them.. Simple directory services are available.

Technical details: See www.globus.org/mds for more information.

Future plans: We need to: expand greatly the set of sensors supported, to include, for example, sensors for catalogs. Hand in hand with the development of additional sensors, we need to define information schemas to facilitate the publication and interchange of data generated by these sensors. With regard to the basic protocols, they need to be extended to provide push-mode as well as pull-mode inquiry. We also plan to look at mapping the information protocols onto different transport mechanisms such as SOAP. We will develop more specialized information indexes with an initial focus on “archiving indexes” and indexes optimized for a variety of query types, including relational queries. Finally, we need to develop access control further to support CAS.

4.4 Resource Management

Resource management interfaces are need to reserve, allocate, and manage the underlying resources that are accessible to the data grid. The different types of data-grid resources are described in the following section. We adopt the Grid Resource Allocation Management (GRAM) protocol as the method for communicating with resources. GRAM defines protocol operations for resource allocation, as well as monitoring and controlling the allocated resource. Resource requests are made via a standard resource specification language (RSL). Current GRAM protocols focus on the management

Current status: GRAM 1.5 provides a reliable interface to computational resources. C, Java, and Python API bindings exist. GRAM 1.5 improves on previous GRAM implementations by providing more robust failure recovery mechanisms and additional RSL attributes, thanks to contributions from the U.Wisconsin Condor group. Limited support for advanced reservation and for additional resource types exists in prototype form [15, 24, 25].

Technical details: See www.globus.org/gram for more information.

Future plans: A new version of the GRAM protocol (GRAM 2.0) is being designed. The focus of this protocol revision is to provide greatly increased robustness and scalability through the use of

soft-state protocols and hierarchical resource management contexts. Additional features will include the ability to specify and enforce a wide range of delegated resource management policies, and support a variety of flexibly resource specification notations.

5 Data Grid Resources

We now define the primary resources with which we are concerned in a Data Grid environment. We consider five classes of resources: storage systems, compute systems, networks, code repositories, and catalogs.

For each, we define the protocols used to access them and the behaviors that we expect these resources to provide. In general, each resource may speak three general classes of protocol:

- *Information protocols*, used to notify the data grid environment of its existence and to allow others to determine its status and configuration;
- *Management protocols*, used to control the behavior of the resource, for example by making reservations; and
- *Access protocols*, used to initiate and control resource use.

Current status: As described above, the Globus toolkit defines GRRP and GRIP as the basic protocols for providing information. In the current toolkit implementation, resource management protocols are restricted to prototype implementations (i.e. GARA), while resource access is supported via GRAM (for computing) and GridFTP (for data).

Technical details: As discussed above, see www.globus.org/mds for information about information protocols and www.globus.org/gram for details on management and access protocols.

Future plans: The GRAM 2.0 protocol will support integrated management and access of resources within the context of a single protocol framework. In addition, information and resource protocols will most likely migrate towards a common, soft-state notification protocol derived from the ideas found in GRRP.

5.1 Storage Systems

We define a storage system not as a physical device but as an abstract entity whose behavior can be defined in terms of its ability to store and retrieve named entities called files. Various instantiations of a storage service differ according to the range of requests that they support (e.g., one might support reservation mechanisms, another might not) and according to their internal behaviors (e.g., a dumb storage system might simply put files on the first available disk, while a smart storage system might use RAID techniques and stripe files for high performance on observed access patterns). They can differ according to their physical architecture (e.g., disk farm vs. hierarchical storage system) and expected role in a Data Grid (e.g., fast temporary online storage vs. archival storage). They can also differ according to the local policies that govern who are allowed to use them and their local space allocation policies. For example, an “exclusive” storage service might guarantee that files are created or deleted, and space reserved, only as a result of external requests; while in the case of a “shared” resource these guarantees might not hold. A related issue might be whether a storage system guarantees that files are not deleted while in use.

We believe that within the context of a Data Grid architecture, it is important to reach consensus on a small set of “standard” storage system behaviors, with each standard behavior being characterized by a set of required and optional capabilities.



Regardless of the type of storage system, we can establish base line capabilities that any storage system should have. These required capabilities are:

- Store, retrieve, and delete named sequences of bytes (i.e., files). Names may be hierarchal, as in the case of typical disk based storage systems, or flat, as is sometimes found in archival storage systems that utilize tape volume identifiers to identify files.
- Provide mechanisms for controlling what operations can be performed on files and the storage system (access control)
- Report on basic characteristics of the storage system, such as total storage capacity and available space.

Additionally, a storage system may support a number of basic optional capabilities including:

- Reserve space for a file.
- Reserve disk bandwidth (guaranteed transfer rate).
- Support a variety of management operations including initiation of high performance (i.e. parallel) transfer, explicit caching or staging of data, etc.
- Ability to report on more detailed attributes such as supported transfer rate, latency for file access (on average or for a specific file), etc.
- Co-located compute resources (i.e., this storage element may support “local” access mechanisms, such as Unix read/write operations).

In addition to the basic functional aspects of storage systems, the *policy* under which a storage system is operated will have a significant impact on how it can be used. One policy element typically found in storage systems concerns the amount of storage that may be used by any one user or application. Specifically, many storage systems support the use of fine-grained quotas (i.e., available space may be larger then the largest file a specific user may be able to store). Storage systems can also be distinguished by the policy that they enforce with respect to the assurances they make with respect to the *time duration* for which they will store files. Specifically, we can identify the following of storage policies:

Finite storage duration. In this situation local operational policy does not guarantee longevity of data. Scratch storage and network based storage element (data caches) fall into this category. In storage systems that implement this policy, additional optional behaviors include:

- Guarantee that files are not deleted while a transfer is in progress.
- Accept hints about desired data lifetimes (e.g., pinning)

Arbitrary storage duration: At the other end of the spectrum are storage systems that provide open-ended storage of files. In its most general formulation, this can result in a storage system with potentially unbounded requirements for storage capacity. To manage the costs associated with building such a system, frequently, several different types of physical storage (such as disk and tape) are integrated into a single system and files are transparently migrated from one to the other based on requests. On such hierarchal mass storage systems, specialized optional behaviors may include:

- Request staging to reduce latency for access to specific files
- Guarantee that files are not migrated while a transfer is in progress
- Accept hints about usage patterns to aid in managing higher levels of the storage hierarchy

- Accept reservations on various levels of the storage hierarchy.

Note that if explicit migration between physical storage types is supported, the storage system can be modeled as a collection of separate storage elements, each with its own management policy and performance characteristics.

By combining different storage systems and policies, a variety of different higher level storage services can be created. For example the hierarchical storage resource manager operations [Shoshani, 1998 #1553] can be defined in terms of a set of storage systems, some representing tape archives (operated under arbitrary storage duration policies) and associated sets of “cache” storage systems, operated under finite storage duration policy. HRM protocol operations such as those proposed in [Bird, 2001 #1758] can then be implemented in terms of basic storage system protocols as applied to the well defined configuration of elemental storage systems and policies.

Current status: GridFTP is available as a standard storage interface, and MDS interfaces to storage systems have been developed, although they require further development. At LBNL, work continues on development of a Storage Resource Manager (SRM) [26] that can provide additional QoS guarantees to clients.

Technical details: www.globus.org/datagrid has more information on GridFTP.

Future plans: Develop standard object classes for characterizing storage systems in terms such as those listed above. Develop storage systems that support reservations. Develop layered protocol specifications for providing integrated support for SRM functionality on top of basic storage system services.

5.2 Compute System

We define a compute system not as a physical device but as an abstract entity whose behavior can be defined in terms of its ability to execute programs. Various instantiations of a compute service will vary along such dimensions as their aggregate capabilities (CPU, memory, internal and external networking) and the quality of service guarantees that they are able to provide (e.g., batch scheduled, opportunistic, reservation, failover, co-allocation of multiple resources). They may also vary in their underlying architecture (uniprocessor, supercomputer, cluster, Condor pool, Internet computing system), although as with a storage service, these details should be of less importance to users than the behaviors that are supported.

As with storage systems, we are interested in defining standard behaviors. Here are our candidates:

Compute farm: On-demand access to computing for single-processor jobs, with or without reservation.

Parallel computer: On-demand access to computing for multi-processor jobs, with or without reservation.

Current status: We use the Globus Toolkit’s GRAM protocol and API to obtain access to remote compute systems, with extensions developed by the Wisconsin Condor group providing enhanced reliability.

Future plans: Develop GRAM-2 to provide enhanced reliability and scalability. Integrate dynamic account allocation mechanisms as a means of providing protection without requiring pre-existing accounts for all users.

5.3 Network

Networks are a critical Data Grid resource, and the ability to discover, understand, predict, and (ideally) manage their performance is critical to our ability to meet application performance goals. In more detail:

- We need to be able to determine the capacity of a network and determine its current utilization. Real-time monitoring is required for this purpose.
- We need to be able to detect and diagnose performance problems. While this requirement arises for any Data Grid component, the distributed and often “black box” nature of networks makes it particularly important.
- We would like to be able to predict expected future utilization.
- We would like to be able to manage the allocation of bandwidth to different purposes.

Each of these requirements is important, but only the first two seem to be on the critical path for Data Grid deployment.

Current status: We use Network Weather Service (NWS) [31] to measure network performance and provide some short-term predictions. This is integrated into MDS. Various tools exist for determining trends and diagnosing performance problems: e.g., Pinger and Pchar. Quality of service mechanisms are addressed by the GARA bandwidth broker and advance reservation system [15], but are not supported in today’s networks.

Technical details: Information about Network Weather Service can be found at <http://nws.cs.utk.edu>.

Future plans: Support for managing network bandwidth explicitly as a resource is one of the design goals for the GRAM 2.0 protocol.

5.4 Code Repositories

A code repository is a storage service used to maintain programs, in source and/or executable form. We discuss it as a distinct entity because of the critical role that online access to programs plays in Data Grids and because code repositories introduce some specialized requirements.

Like a storage or computational resource, a code repository should provide information about its contents, as well as providing access to the actual code elements. The capabilities require in a code repositories remain to be defined, but should presumably include the following:

- Ability to manage multiple revisions of the same application as well as versions for different architectures.
- Provide information about code, including complete, platform requirements for execution, performance profiles, and a description of input and output parameters.
- Support for dynamic linking and shared libraries.
- Access control and policy enforcement.

As with compute and storage systems, there are many instantiations of this basic concept (e.g., many physics applications have developed their own code repositories).

Code distribution is a related requirement; we discuss this in Section 9.2.

Current status: The GridCVS utility supports GSI-authenticated access to CVS repositories. Within the Atlas experiment, the PACMAN system

(<http://www.usatlas.bnl.gov/computing/software/pacman/>) is being used for distribution of precompiled software packages (either tarballs or Linux RPMs).

Technical details: See www.globus.org/gridcvcs.

Future plans: We need to collect comprehensive requirements for code repository and distribution services, and either import this capability from elsewhere or develop it ourselves. We also need to consider how to collect, represent, and store performance information used for query estimation. We need to specify the relationship between the Transformation Catalog (TC) [ref], which provides information a mapping between a transformation and its location. TC can also include metadata about the transformation, such as who has designed it, who validated it etc....

5.5 Catalog Services

A catalog is a storage service used to maintain mappings of some sort. We discuss it as a distinct resource because of the critical role that online access to such mappings plays in Data Grids and because catalogs introduce some specialized requirements (see [Holtman, 2001 #1759], for example).

In [9], we identify a need for the following catalog services:

- *Metadata catalog* [4] that handles mappings from “entity id” to “entity attributes” for the entities of interest, which are files in the short term but may be objects in the future.
- A *Grid Container Management Service* (GCMS) will be used to associate objects to logical containers as well as to provide the ability to import/export data into the grid.
- *Replica catalog* [2] (or, as we argue in [5], *replication location service*) that maintains information about the physical location of replicas of files (or containers in the future).
- Other catalogs concerned with supporting transparency with respect to materialization, such as derived data catalogs, meta derived data catalogs, and transformation catalogs [9].

Notice the clear distinction made between metadata and replica catalogs. This distinction is important for three reasons. First, many applications maintain their own metadata catalogs; we do not want to force them to change them to support replica information. Second, we anticipate access patterns on metadata and replica data to be different. Third, as we discuss in more detail in Section 7.3, it is not necessarily vital that replica location information always be completely consistent—while metadata typically must be.

All of these catalogs require appropriate support for access control and policy enforcement. All can have challenging scalability requirements in some Data Grid environments: e.g., LHC experiments talk about 10s of millions of files and 100s of replica sites. In addition, we note that like storage and computational resources, catalogs should provide information about their availability and contents via MDS, as well as providing access to the actual data elements. We discussed this requirement in Section 4.3 also.

In the rest of this section, we discuss requirements for metadata

5.5.1 Metadata Catalog

A metadata catalog maintains a mapping from entity name (e.g., object name or logical file name) to entity attributes. These attributes are an essential part of the entity, and must be maintained and protected with as much care as the data. (In fact, in some cases, metadata may be *more* important: we may be to recreate data but not metadata.) We are aware of the following potential requirements:

- *Security*: access control for reading and writing. May be on a per-attribute basis (?). Integration with a community authorization service will be important.
- *Scalability*: In some domains, the number of entities can be large (certainly tens of millions, perhaps significantly more). We have been told that read and write rates will be low (tens or at most hundreds per second), but we need more information on this point. However, it is expected that reads will dominate the accesses.
- *Persistence*: Metadata needs to be protected against unplanned deletion. To what extent remains unclear.
- *Fault tolerance*: Mirroring of metadata may be important, to ensure accessibility in the event of network partitions or other failures.
- *Extensibility*: We may need to record attributes relating to Data Grid operation, e.g., concerning the location of a “master” copy of a file, the transformations used to generate a piece of derived data, and so forth.

All the above requirements are also important to the DMDC [ref], which provides similar functionality to the MDC, but in the virtual space.

We do not yet have good numerical estimates of performance requirements.

Current status: A variety of implementation options are possible and we have not decided on (indeed may not be able to) decide on a single approach. Metadata catalogs are typically provided by applications, although with additional attributes required for Data Grid operation [9]. In this case, we may simply be concerned with providing for GSI-enabled remote access. It would also be desirable to provide a “standard” metadata solution for use when no other solution is available, although it is not yet clear what that standard should be. SRB [4] uses an SQL-like syntax. The Open Archives Initiative Protocol [27] may be relevant. Ultimately we need to address issues of persistence, fault tolerance, and scalability.

Future plans: Define requirements for; develop; and make available a standard metadata solution.

5.5.2 Replica Catalog

Requirements and current status of work on replica location services are discussed in Section 7.3.

Current Status: we have a design and implementation of a centralized replica catalog, which maps logical collections and logical file names to a particular physical file instance [Allcock, 2001 #1662].

Future plans: a new design is being proposed for enabling a construction of a distributed replica location service.

5.5.3 Transformation Catalog

The transformation catalog will be used to store information about the transformations, which need to be used in order to process data as requested by the user. The catalog can be used to process raw or derived data products. Data derived using the transformations will be possibly placed in the Metadata catalog or the Replica Catalog.

A major factor in deciding on the design is the need for information about the architecture and the platform of the machine where the computation is to be done. Another important factor is whether a binary is available or not for that architecture. If binaries are available the shared libraries are assumed to be included in the tarball distribution as they may be required to run the executable. We also need to provide compiler information (which compiler was used to generate the code) as well as the flags used. If the source is distributed we need to provide a means for describing how

to build the software. Factors such as cost of running or accessing the transformation as well as the minimum resource requirements have to be taken into consideration.

Current Status: Several prototype transformation catalogs have been designed and are being evaluated within the context of CMS and LIGO experiments.

Future Plan: Discuss the current design and understand the relationship to the code repositories.

5.5.4 Virtual Data Catalogs

Work is under way to develop requirements for, and explore via prototypes, the virtual data management catalogs, such as the DMDC and the DDC [Deelman, 2001 #1663]. The derived data catalog (DDC) records information associated with derived datasets, such as the transformation needed to generate the derived dataset, the cost of that transformation, and a unique derived dataset name. DDC deals with derived datasets at the level of data objects. “Meta attributes” of the catalog entries derived data, such as owner, name, author, and creation-time are also recorded here. This element together with the DMDC (below) and the transformation catalog provide information required to support the manipulation of derived data.

The derived metadata catalog (DMDC) is similar to the MDC but it maps from a set of application-specific attributes to a derived data object id(s) that indexes into the DDC.

The difficulty is in the understanding of how to name virtual data products and how to describe the materialization process. One idea is to integrate the G-DAG ideas and structures described in Section 3. For example, in the DDC, we might want to describe the process of obtaining the virtual data using a G-DAG description.

5.5.5 Other Catalogs

Work is under way to develop requirements for, and explore via prototypes, the other catalogs referred to above. More detail will be provided in a subsequent version of this document.

Current status: There is a document [Deelman, 2001 #1663] that specifies the structure of the virtual data catalogs.

Future plans: Evaluate the design of the catalogs using virtual data as described by the experiments.

6 Request Planning and Execution Services

The first higher-level services that we discuss are those concerned with request planning and execution. From the perspective of our Data Grid architecture, we must:

1. Define or use established protocols for the request management services to interact with the catalog and information services
2. Define a common representation for the (typically partially ordered) sets of tasks or operations that must be performed on the Data Grid, and
3. Define a protocol for submitting task descriptions using the above representation to planner and execution services. We must also define protocol messages for monitoring and controlling progress as these services operate on the submitted tasks.
4. Define protocols and policies for the execution services to operate in case of failure.

Many different representations of task sets exist and could be considered for our purposes. However, we propose to represent data grid tasks via a graphical data-flow-type notation that models the operations (nodes) and the flow of data (arcs) and that defines one or more triggering

rules (all available, N out of M available, etc). Data-flow concepts have proven to be useful in a variety of workflow, scripting, and composition applications (cite AVS, workflow) and we believe that they are powerful enough to represent the types of operational activities that need to be performed in the Data Grid context.

6.1 Abstract and Concrete Representations

Before a set of operations can be executed on a data grid, it must be completely instantiated: storage systems and physical file name specified, specific computational resources identified, specific versions and implementations of applications located, etc. We refer to a plan that has this level of detail to be a *concrete* plan.

On the other hand, a task graph specified by a user might be expressed in terms of high-level, abstract operations, specifying, for example, names of programs and not specific versions, logical file names, data attributes, etc. We refer to such a plan as *abstract*.

End users will tend to specify abstract plans. Virtual data requests by definition will be abstract. Execution services will expect to see concrete plans. At various points in the process of manipulating a plan, it may contain a mixture of concrete and abstract elements. The planning and execution process will perform various activities designed to translate abstract plans into concrete plans.

6.2 Grid Directed Acyclic Graphs

Many different data-flow representations have been proposed: each with a different set of basic nodes and associated semantics. Nevertheless, because they can be mapped into an underlying graph model, it is possible to create a common graph-based syntax to represent these different semantics.

While we admit the existence of alternative representations of task sets, we propose to use a specific simple representation for the initial version of the Data Grid architecture. Specifically, we will represent data grid computations as Grid directed acyclic graphs (G-DAGs).

To discuss: how does this relate to various JCLs that are being discussed, e.g. in PPDG?

6.2.1 G-DAG Syntax

A directed acyclic graph (DAG) comprises a set of nodes connected by directed edges. Both nodes and edges may be labeled. We can represent a DAG graphically or in XML, as shown in Figure X. (See Appendix X for more details.) I propose that we define an XML syntax, I think that is the right thing to do. But let's look at what XML tools already exist first. What about choreography tools? We should also define Miron's syntax



Figure 4: From left to right, graphical, Condor, and XML representations of a directed acyclic graph with four nodes.

6.2.1.1 DAGMan syntax

As a starting point we propose that the syntax used by the Condor system's Directed Acyclic Graph Manager (DAGMan). DAGMan is a meta-scheduler that sequences the submission of activities to Grid resources, submits jobs in an order represented by a DAG and processing the results.

A DAG description consist of:

1. A list of the activities (i.e. programs) in the DAG.
2. Pre/post processing that takes place before/after the submission of any programs in the DAG to a Grid resource.
3. Description of the dependencies in the DAG.

The following is an example of a DAG that implements the diamond graph shown above (http://www.cs.wisc.edu/condor/manual/v6.2/2_10Inter_job_Dependencies.html):

```
Job A A.condor
Job B B.condor
Job C C.condor
Job D D.condor
Script PRE A top_pre.csh
Script POST C mid_post.perl $JOB $RETURN
PARENT A CHILD B C
PARENT B C CHILD D
```

The first 4 lines specify the jobs and the scripts that contain the job specification. Additionally, the file specifies a pre-processing for job A and a post-processing for job C (next two lines). Finally, the dependencies between the task as shown in Figure 4 are specified in the last two lines.

6.2.2 G-DAG Semantics

We interpret a DAG as follows. Nodes represent tasks to be performed. Edges represent dependencies between tasks and are typically labeled with the name(s) of dataset(s) produced by the predecessor task and consumed by the successor task. The semantics of a G-DAG are as follows: if two tasks are connected by an edge, then the predecessor must complete execution successfully before the successor can be executed. Notice that this semantics prevents pipelining of tasks; we will need to see if it is too restrictive.

The proposed initial DAG semantics requires that if any task fails, then the entire G-DAG fails. If failure recovery is to be possible, these semantics require that each node in the G-DAG exhibit and all or nothing transactional behavior with respect to side effects. Upon failure, it may be possible to resubmit the partially completed G-DAG (restart) or to restructure the G-DAG to use alternative resources (fail-over). Note that in general, it would be desirable to augment G-DAG semantics to specify alternative execution paths as part of the execution plan.

As defined above, a plan and hence a G-DAG can be either an *abstract DAG* and a DAG that only includes references to physical files: a *concrete DAG*. As illustrated in Figure 5, we can think of request planning operations as progressively rewriting an abstract DAG to an increasingly concrete form. On the left, we see a request for a value V. This has not been materialized, so the next step is to determine the computation required to produce it, hence producing the DAG in the center. The final planning phase identifies the physical files and locations that will be used to perform the computation, as encoded in the DAG on the right.

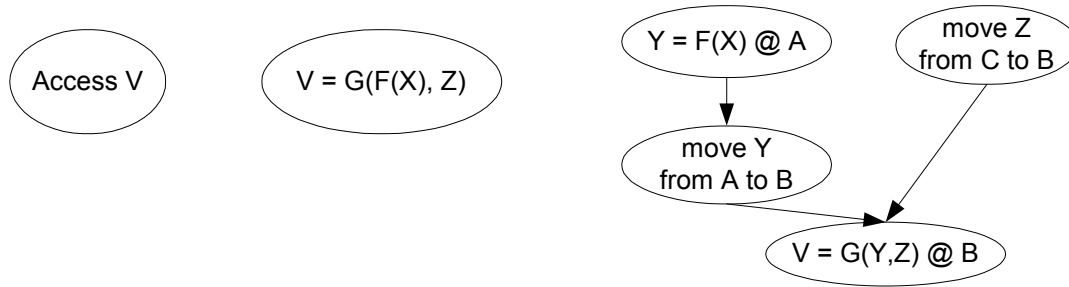


Figure 5: Rewriting a G-DAG from, on the left, an abstract request (access a value V) to a computational request (center) and hence to a concrete set of computations and data movement operations (on the right). V , X , Y , and Z are data values, and A , B , and C are sites.

The description of the request planning process in the preceding paragraph is likely to be overly simplistic, in that it does not allow for decisions made dynamically during execution. Dynamic decisions may be needed for several reasons. First, the data consumed by a computation may be determined only during its execution. Second, it may be desirable to delay decisions concerning where to execute tasks and/or which copies of data to access until execution time. Hence, in the terms of Figure 1, the DAG passed to the request planner may not be totally concrete. We anticipate that our G-DAG syntax and semantics will evolve over time as we learn more about these issues.

6.2.3 Planned Extensions

It is our intention to extend the G-DAG syntax and semantics in a number of areas, including the following:

- *Query estimation:* We want to allow nodes (and edges?) to be annotated with estimated costs, for use during the planning process.
- *Alternative plans:* We want to allow nodes (and edges?) to be annotated with information concerning what to do in the case of failure.
- *Client interface:* We want to allow the insertion of additional nodes, which can communicate with the client and provide information such as execution status.

These extensions are topics of current investigation.

Current Status. We use the Condor DAGMAN toolkit [20] as our G-DAG reference implementation. The system supports simple graph description in which one specifies the precedence relationship between named nodes, and the programs to be run at each node and local pre and post node processing.

Future plans: While DAGMan provides a reasonable starting point for G-DAG notation and semantics, the discussion above illustrates a number of enhancements that we believe will need to be made. These include:

- Create an XML based representation of DAG structure. Investigate, for example, RDF [1], although this may be overkill.
- Extension of DAG structure to support additional attributions such as cost annotations and link annotations to support pipelining
- Extend DAG structure to accommodate error handling extensions and alternative execution paths.

- Investigate alternative dataflow structure such as those being considered for workflow applications.

6.3 Request Planning Services

Request planning issues are not especially well understood at this point in time, but are the topic of much current research in various application projects. Our current understanding of the problem is that:

- The user provides a request as a G-DAG that refers to various data values without regard to their location or materialization.
- An entity called a *request planner* consults various catalogs and information sources to determine what data exists, and what computations and data movements may be required to process the request. For example:
 - If the data is materialized, the planner evaluates the costs of accessing the data from different locations. We might also evaluate the cost of re-computing the data, in case it is cheaper to do so.
 - If the data is not materialized the various derived data catalogs are consulted to establish how to produce the data and the cost of doing so.
- Information services are consulted to establish resource availability and estimate the performance costs of data evaluation and retrieval.
- The output of this process (a plan or plans) is a more detailed G-DAG.
- Finally, it might be desirable to allow the user/application to decide whether, given the estimated cost, they want to proceed with data materialization or which way to proceed, batch vs. interactive for example.

Current status: We do not have any generic request planning capabilities. However application-specific request planners for LIGO and CMS have been developed. In the LIGO prototype being currently developed, the user specifies the request in XML format (or via a GUI interface, which produces an XML request). The request planner identifies which data products need to be produced and design a plan in DAGMan's format.

Future plans: Develop request planning activities. Look into agent technologies as a possible framework. Although some of the planning is expected to be application-specific, we need to look at components, which are generic, such as finding the "best replica", "find appropriate compute/storage resources", etc.

6.4 Request Execution Services

The request execution services need to be able to take a detailed G-DAG description and execute the components on a specified resource. The components can represent data movement, computations, access to replica management services, etc....

The request execution services need to be able to be integrated with the request planner in order to deal with task execution failures. Consider the situation where the request planner has determined via the Replica Location Services (see Section 7.3) that a given file is available at locations X and Y. Using a replica selection algorithm, the planner decides to make a plan to access the data at location X. It is possible however, that by the time the request executor is accessing the data, the data is no longer present at location X. In that case, we have two choices:

1. the request planner can enough information for the request executor to be able to find another copy of the data

2. interface the planner to the executor in such a way, that in the case of a node execution failure, the executor can ask the planner for an alternative plan.

Requirements: ???

Our initial execution service is based on DAGMan and Condor-G. The syntax of a DAGMan plan was described above. DAGMan supports very simple error semantics. If any node in the DAG fails (as indicated by its return code), the entire DAG is aborted after all other independent nodes finish. Instead of resubmitting the jobs, DAGMan generates a “Rescue DAG,” that is functionally the same as the original DAG file, but it includes indication of successfully completed nodes. If the DAG is resubmitted, the jobs marked as completed will not be resubmitted. This Rescue DAG is automatically generated by DAGMan when a node within the DAG fails.

Request execution involves three distinct activities:

- *Request execution*: The needed resources are allocated (if possible) and the computation is executed. The Condor DAGMan represents the current state of the art in this area, providing required services such as credential refresh and reliable restart in the event of failure. See [16] for some details.
- *Request monitoring*: We need to be able to monitor the progress of the request so that it is available to the user/application. The request execution service needs to be able to deal with failures and construct contingency plans. This functionality is currently part of DAGMAN. However, we need an interface from DAGMan to the application to make the desired monitoring information available.
- *Error recovery*: We need to be able to recover from simple failures, such as unavailability of the data at a given location.
- *Information update*: Upon request completion, we need to notify and return the output to the user/application. We also need to update catalogs to reflect that the data has been materialized. Just how and when these updates occur is not fully understood. For example, if a transformation produces a whole range of data values, we might not want to register them all, but instead discard them. To decide this issue we need to enhance our understanding of how the application programs (transformations in our model) behave.

Current status: The DAGMan system supports the reliable execution of concrete G-DAGs containing both computational and data movement tasks. DAGMan is used in conjunction with Condor-G to provide the ability to refresh a users credential in the case of long running jobs and to supports access to GRAM- and GridFTP-mediated resources (i.e., computers and storage systems)

Future plans: Extend DAGMan to support more dynamic execution plans including plans with input/output sets that are determined at runtime, alternative execution paths and user-specified error recovery strategies. Explore and implement catalog update procedures. Define a network protocol to enable DAGMan to be run as a GSI-authenticated remote service. Define protocols and interactions between the request executor and the request planner and their relative responsibilities. Define the model for error recovery.

7 Information Services

We use the term information services to refer collectively to services concerned with resource discovery and monitoring.

7.1 Discovery and Monitoring Infrastructure

An information service supports enquiries concerning the structure, state, availability, etc., of multiple Grid resources. Resource attributes may be relatively static (e.g., machine type, operating system version, number of processors) or dynamic (e.g., available disk space, available processors, network load, CPU and I/O load, rate of accomplishing work, queue status and progress-metrics for batch jobs). Different information service structures may be used depending on the types of information to be queried and the types of queries to be supported. For example, NetLogger [30] is frequently used for application monitoring.

Current status: We propose to adopt the Globus Toolkit's Meta Directory Service (MDS) information service architecture. MDS uses the GRRP and LDAP registration and enquiry protocols to construct Grid Index Information Services (GIISs) that receive GRRP registration messages and maintain a list of active services; use LDAP queries to retrieve resource descriptions, from which they construct indices, with time to live information indicating how frequently indices should be updated; and use those indices to process incoming LDAP queries.

Future plans: Investigate further the GIIS functionality required in Data Grid applications. Develop standard object classes for additional entities. Address issues of archiving of monitoring information.

7.2 Collective Monitoring Services

The broad deployment of standard information access protocols such as those provided by MDS makes it feasible to construct a wide variety of monitoring systems that can perform various sorts of anomaly detection and system characterization. However, little work has been done in this area to date.

Intrusion detection. Techniques will clearly be required for monitoring the ensemble of Data Grid resources to detect coordinated attacks [10, 22].

Anomaly detection. Techniques are required for monitoring the ensemble of sensors associated with a Data Grid to detect various types of anomalies, such as network performance problems, scheduling problems, and attacks. Having detected an anomaly, such systems can also potentially attempt to correct detected problems.

Resource characterization. Monitoring can also be used to develop summaries of resource characteristics, relating for example to relative quality and availability.

7.3 Replica Location Service

Efficient, reliable, and secure access to and management of information about the location of file replicas is an important service in a Data Grid. We discuss the design and implementation of this Data Grid component in other documents.

Current status: A replica catalog API has been designed and a centralized implementation constructed; the implementation is being used in a variety of Data Grid applications and is proving sufficient for short-term requirements [2]. A requirements analysis and design for a more scalable, distributed replica location service has been developed [5].

Future plans: Develop a scalable, distributed replica location service.

7.4 Catalog Management Services

The issue of catalog management is complex, as the catalogs deal with both application-specific and data-grid specific entities. For example, both the derived metadata catalog (DMDC) and the MDC defined in [9] contain application-specific attributes. Applications have spent considerable

effort to develop MDCs, and it is not within the scope of this work to design the catalog schemas for applications: instead, we want to design interfaces to existing structures. Therefore it is necessary for the Data Grid architecture to:

- Utilize currently existing MDCs. Provide means of querying and updating the catalogs.
- Provide a derived data catalog (DDC) structure that captures the descriptions of transformations performed by the applications.
- Build DDC and replica catalog (RC) structures that are as generic as possible.
- Provide well-defined interfaces to the catalogs, so that both users and applications can construct relevant queries.

Current status: Initial proposal for the virtual data catalogs has been proposed [9], but needs to be further evaluated and discussed.

Future plans:

- Design mediators that can interface with the application-specific catalogs. These mediators will allow the Data Grid to retrieve information from the various catalogs already developed by the applications as well as update those catalogs upon data materialization.
- Augment the databases used by the applications with new attributes that support the Data Grid, such as the transformation used to produce the data, in the case of MDC or the ids of the derived data product, in the case of DMDC.
- In the DDC, support attributes that are common to all applications, such as transformation names and input file names. Explore the use of G-DAGs as derived data descriptors in the DDC.

7.5 **Grid Container Management Services(GCMS)**

GCMS will be used to associate objects to logical containers as well as to provide the ability to import/export data into the grid.

Importing into the grid is performed by writing objects and their associated attributes into a *grid container*. Accessing of objects is done via the export function of GCMS. Obviously, GCMS is an application specific component, as it needs to understand the structure of an application object and it needs to decide on the best mapping of objects to containers for a given application.

Current Status: GCMS is still in the design stage.

- Future Plans: We need to fully understand what are the requirements on that service. Look into SRB and see how it uses containers to manage collections.

8 **Policy and Security Services**

The GSI mechanisms described in Section 4.2 address basic authentication and authorization requirements. A set of higher-level services build on this base to address issues of policy and credential management that arise in a Grid environment. Specifically:

- The *community authorization service* supports the specification and enforcement of the *community policies* that determine who is allowed to use what resources for what purposes.

- *Online credential repositories and online certificate authorities* address scalability and usability issues associated with public key infrastructure.

8.1 Community Authorization Service

Community authorization services [ref? – at least refer to Laura’s URL for the CAS2 doc] (CAS) are an important component of the Data Grid authorization architecture. A CAS enables resources used by a community to implement a shared global policy for use of these community resources. Examples include enabling group access to a specific collection of files (regardless of where they are stored), or ensuring fair share use of network bandwidth across all members of the community. The basic idea is to enable a resource owner to delegate authorization decisions to a community representative (the CAS) that specifies and enforces the global policies, such as “ingestion of new data takes precedence over analysis” or “no one ATLAS user may consume more than 30% of the compute resources dedicated to ATLAS.”

A CAS architecture is under development and will be described in a separate document. This CAS definition leverages GSI security protocols. The central design concept is to use the CAS to issue credentials to a user that are good for a class of operations, and for other services to be able to interpret the validity of the requested operation without resorting to any other on-line service.

Current status: An early CAS prototype was demonstrated in August 2001. Development is ongoing, and a more concrete version will be available by end of 2001.

Future plans: Complete CAS development and integrate CAS capabilities into data and compute servers.

8.2 PKI Scalability and Usability

An issue of ongoing investigation within the Globus project (under DOE funding) relates to the development of techniques aimed at increasing scalability and usability of public key infrastructure (PKI) systems. For example, online certificate authorities can be used to map from local credentials (e.g., Kerberos tickets) to PKI credentials. Online credential repositories can be used to cache credentials for subsequent use from e.g. browsers [23]. These developments are not immediately on the critical path for Data Grid systems, but will be important long term.

Current status: An online credential repository, MyProxy, exists [23].

Future plans: Track ongoing work in Globus project and GGF.

9 Replication

We discuss two issues here: data replication, including event data and metadata catalogs, and code distribution.

9.1 Replica Management Services

An effective technique for improving access speeds and reducing network loads when many users must access the same large datasets can be to replicate frequently accessed datasets at locations chosen to be “near” the eventual users. However, organizing such replication so that it is both reliable and efficient can be a challenging problem, for a variety of reasons. The datasets to be moved can be large, so issues of network performance and fault tolerance become important. The individual locations at which replicas may be placed can have different performance characteristics, in which case users (or higher-level tools) may want to be able to discover these characteristics and use this information to guide replica selection. And different locations may have different access control policies that need to be respected.

A replica management service is an entity responsible for keeping track of replicas, providing access to replicas, generating (or deleting) replicas when required. This service can clearly layer on functionality described above, specifically the Replica Catalog and Storage System access protocols and reliable high speed data movement.

We note that high-level storage management functions can be created at the collective level by composing elemental storage systems via resource level protocols. For example, hierarchal storage management can be viewed as a replication policy that manages an archival storage element and a networked storage element.

Numerous other replica management strategies can be imagined, but the following basic mechanisms appear to have general utility:

- Reliable, high-speed data movement – Ruth: I still think this is a separate service – Data Transport –above communications and below Replica Management.
- Fault tolerant and scalable Replica catalogs able to keep track of where replicas have been created and “in progress” replication efforts.
- Mechanisms for creating new replicas and removing old ones
- Mechanisms for checking on and/or enforcing the consistency of existing replicas. (How can we do this without knowledge of the internal structure of files?)

Current status: We have in place replica management functions developed within the Globus Data Grid Toolkit [2] and within the Storage Resource Broker system [<http://www.npaci.edu/DICE/SRB/index.html>]. Both systems support multiple data transfer protocols. These functions build on the GridFTP or other data transport protocols or application services and replica catalog mechanisms described below to provide functions for:

- The registration of files with the replica management service.
- The creation and deletion of replicas for previously registered files.
- Inquiries concerning the location and performance characteristics of replicas.
- The updating of replicas to preserve consistency when a replica is modified. (A formal definition of consistency in the Replica management services is yet to be defined).
- Management of access control at both a global and local level.

We also have in place the services provided by GDMP [18, 28], although these are very specific to a particular implementation. For example, GDMP assumes the use of Objectivity databases.. More. Other data transport services such as bbcp (<http://www.ihep.ac.cn/~chep01/paper/7-018.pdf>), bbftp (<http://doc.in2p3.fr/bbftp/>) are used by several experiments. These provide short term enhanced services tailored for the HENP environment.

Technical details: Pointers to be provided.

Future plans:

- Determine experimentally how effective these techniques are in practice.
- Create a *replica management service* that encapsulates these functions and supports remote requests for replica creation, etc. (A prototype is to be shown at SC’2001 in November.)
- Develop basic ideas for replica generation/deletion services. Work is underway at U.Chicago in this area, by PhD student Kavitha Ranganathan.

9.2 Code Distribution

Secure, reliable code distribution is a critical concern for Data Grids: it must be easy to distribute a new piece of application software to a set of compute sites, and to update software as and when required. The problem is clearly complex and multidimensional. Publication of software installation information via MDS is important.

Current status: All physics experiments have some technologies and a certain degree of automation to support code distribution and verification BaBar uses cvs and rshell distribution scripts; D0 uses cvs and the Fermilab ups/upd utilities; CMS uses SCRAM which is being extended to support code distribution; ATLAS uses CMT and PACMAN. EU DataGrid are looking at LCFG [3] from Edinburgh, which we should track. Globus publishes information about installed Globus software (but not other software) via MDS.

To do: Refine requirements statement. Survey available software. Enlist someone to produce something.

10 Performance Estimation and Evaluation

11 Planning, Execution and Error Recovery

12 Missing Components

For convenience we provide here a list of the major Data Grid components identified in this document for which no implementation exists or is expected to become available in the near future. These components represent opportunities for others to contribute to development.

Code distribution service. See Section 9.2.

Intrusion detection service; activity logging. See Section 7.2.

Accounting. See Section X.

Debugging and tracing.

To do: Add to the list of missing services.

Acknowledgments

We are grateful to our colleagues within the Earth Systems Grid, European Data Grid, GriPhyN, and Particle Physics Data Grid projects for numerous helpful discussions on the topics presented here. We acknowledge, in particular Ewa Deelman and Mike Wilde for comments. This work was supported by the GriPhyN project under contract XXX and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Bibliography

1. RDF. <http://www.w3.org/TR/REC-rdf-syntax/>.
2. Allcock, W., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S., Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In *Mass Storage Conference*, (2001)

3. Anderson, P. and Scobie, A., Large Scale Linux Configuration with LCFG. In *4th Annual Linux Showcase and Conference*, (2000).
www.usenix.org/publications/library/proceedings/als2000/full_papers/anderson/anderson.pdf
4. Baru, C., Moore, R., Rajasekar, A. and Wan, M., The SDSC Storage Resource Broker. In *Proc. CASCON'98 Conference*, (1998)
5. Chervenak, A., Foster, I., Iamnitchi, A. and others. A Scalable Replica Location Service. Work in progress., 2001.
6. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *J. Network and Computer Applications* (23). 187-200. 2001.
7. Collaboration, T.G. The GriPhyN Project. www.griphyn.orgTBD, 2000.
8. Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C., Grid Information Services for Distributed Resource Sharing. In *10th IEEE International Symposium on High Performance Distributed Computing*, (2001), IEEE Press, 181-184
9. Deelman, E., Foster, I., Kesselman, C. and Livny, M. Representing Virtual Data: A Catalog Architecture for Location and Materialization Transparency. 2001.
10. Forrest, S., Hofmeyr, S.A. and Somayaji, A. Computer Immunology. *Communications of the ACM*, 40 (10). 88-96. 1997.
11. Foster, I. and Kesselman, C. A Data Grid Reference Architecture. GriPhyN 2001-6, 2001, www.griphyn.org/document-server.
12. Foster, I. and Kesselman, C. (eds.). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.
13. Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A Security Architecture for Computational Grids. In *ACM Conference on Computers and Security*, 1998, 83-91.
14. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15 (3). 200-222. 2001. www.globus.org/research/papers/anatomy.pdf.
15. Foster, I., Roy, A. and Sander, V., A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation. In *Proc. 8th International Workshop on Quality of Service*, (2000)
16. Frey, J., Tannenbaum, T., Foster, I., Livny, M. and Tuecke, S., Condor-G: A Computation Management Agent for Multi-Institutional Grids. In *10th International Symposium on High Performance Distributed Computing*, (2001), IEEE Press, 55-66
17. Gullapalli, S., Czajkowski, K., Kesselman, C. and Fitzgerald, S. The Grid Notification Framework. Global Grid Forum, Draft GWD-GIS-019, 2001.
18. Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H. and Stockinger, K., Data Management in an International Data Grid Project. In *International Workshop on Grid Computing*, (2000), Springer Verlag Press
19. Litzkow, M. and Livny, M. Experience With The Condor Distributed Batch System. In *IEEE Workshop on Experimental Distributed Systems*, 1990.
20. Livny, M. DAGMAN reference.
21. Moore, R., Baru, C., Marciano, R., Rajasekar, A. and Wan, M. Data-Intensive Computing. In Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 105-129.
22. Mukherjee, B., Heberlein, L.T. and Levitt, K.N. Network Intrusion Detection. *IEEE Network*, 8 (3). 26-41. 1994.
23. Novotny, J., Tuecke, S. and Welch, V., An Online Credential Repository for the Grid: MyProxy. In *10th IEEE International Symposium on High Performance Distributed Computing*, (2001), IEEE Press, 104-111

24. Sander, V., Adamson, W., Foster, I. and Roy, A., End-to-End Provision of Policy Information for Network QoS. In *10th IEEE International Symposium on High Performance Distributed Computing*, (2001), IEEE Press, 115-126
25. Sander, V., Foster, I., Roy, A. and Winkler, L., A Differentiated Services Implementation for High-Performance TCP Flows. In *TERENA Networking Conference*, (2000)
26. Shoshani, A., Bernardo, L.M., Nordberg, H., Rotem, D. and Sim, A. Storage Management for High Energy Physics Applications. In *Computing in High Energy Physics 1998 (CHEP 98)*, 1998.
27. Sompel, H.V.d. and Lagoze, C. The Open Archives Initiative Protocol for Metadata Harvesting. The Open Archives Initiative, 2001, http://www.openarchives.org/OAI_protocol/openarchivesprotocol.html.
28. Stockinger, H., Samar, A., Allcock, W., Foster, I., Holtman, K. and Tierney, B., File and Object Replication in Data Grids. In *10th IEEE Intl. Symp. on High Performance Distributed Computing*, (2001), IEEE Press, 76-86
29. Szalay, A. and Gray, J. The World-Wide Telescope. *Science*, 293. 2037-2040. 2001.
30. Tierney, B., Johnston, W., Crowley, B., Hoo, G., Brooks, C. and Gunter, D., The NetLogger Methodology for High Performance Distributed Systems Performance Analysis. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, (1998)
31. Wolski, R. Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, Portland, Oregon, 1997.